PETRE STOICA IOAN TĂBUȘ
MIHAI TERTIȘCO

METODE RAPIDE PENTRU MODELAREA SEMNALELOR ȘI SISTEMELOR

Editura Academiei Române

Scanned with OKEN Scanner

Redactor : EUGEN MARINEANU

Bun de tipar : 14.02.1997. Format : 8/61 x 86.
Coli de tipar : 28.
C.Z. pentru biblioteci mari 007 : 621.39
621.39 : 007
C.Z. pentru biblioteci mici : 007

Scanned with OKEN Scanner

PETRE STOICA IOAN TĂBUŞ MIHAI TERTIŞCO

METODE RAPIDE PENTRU MODELAREA SEMNALELOR ŞI SISTEMELOR

EDITURA ACADEMIEI ROMÂNE
Bucureşti, 1996

Scanned with OKEN Scanner

Fast Methods for Signal and System Modelling

© Editura Academiei Române
Toate drepturile asupra acestei ediţii
aparţin editurii.

ISBN 973-27-0587-6

Editura Academiei Române
Bucureşti, R-76117, Calea 13 Septembrie nr. 13, Sector 5

Scanned with OKEN Scanner

Prefaţă

Modelarea sistemelor *este denumirea generică pe care o vom utiliza în această lucrare pentru a desemna operaţia de determinare experimentală a unui model matematic pentru un semnal sau sistem. Aplicaţiile "modelării sistemelor" în ştiinţă şi tehnică sunt omniprezente: de la biologie, ecologie şi econometrie la geofizică, comunicaţii, acustică şi automatică. Într-o serie din aceste aplicaţii, modelele implicate au un număr relativ mic de parametri (de exemplu 20 de parametri sau mai puţin). Determinarea unor astfel de modele poate fi realizată utilizând metode standard de modelare şi identificare a sistemelor. Totuşi, o serie de aplicaţii recente importante cer determinarea, deseori* în timp real, *a unor modele descrise de* foarte mulţi parametri *(de exemplu, mai mulţi de 1000 !), deziderat ce necesită metode speciale de modelare a sistemelor. Astfel de aplicaţii apar, de exemplu, în următoarele domenii:*

- *Estimarea densităţii spectrale a semnalelor cu spectru larg.*
- *Filtrarea adaptivă digitală.*
- *Comunicaţii telefonice (de exemplu, anularea ecoului în liniile telefonice).*
- *Acustică şi analiza semnalelor vocale.*
- *Seismologie (de exemplu, deconvoluţie, prospecţiuni geologice utilizând tehnici de reflexie).*
- *Conducerea adaptivă a sistemelor multivariabile.*

Pentru rezolvarea problemelor de modelare asociate, de exemplu, cu aplicaţiile menţionate mai sus, a fost necesară elaborarea unor metode rapide *de modelare a sistemelor. Aceste metode exploatează structurile algebrice particulare ale problemelor de modelare corespunzătoare diverselor aplicaţii, pentru a elimina orice "calcul redundant" efectuat în cadrul metodelor standard. (Un exemplu tipic îl constituie operarea cu matrice Toeplitz, care în algoritmii standard de modelare sunt tratate ca matrice arbitrare, în timp ce metodele rapide exploatează intens structura foarte specială a acestui tip de matrice). Denumirea generică de "metodă rapidă" este utilizată pentru orice algoritm care rezolvă problema formulată de modelare într-un timp mai scurt decât cel necesitat de un algoritm standard. În general, totuşi, metodele rapide nu sunt numai "mai rapide", ci sunt "mult mai rapide" (de exemplu, inversarea cu metode standard a unei matrice Toeplitz de dimensiuni $n \times n$, cere $\mathcal{O}(n^3)$ operaţii aritmetice, în timp ce inversarea rapidă a acesteia se poate efectua în $\mathcal{O}(n^2)$ sau chiar $\mathcal{O}(n \log n)$ operaţii).*

Lucrarea de faţă îşi propune să prezinte într-o manieră unitară cele mai performante şi mai frecvent utilizate metode rapide de modelare a sistemelor. În ţară nu există nici o lucrare similară, prezenta monografie constituind deci o premieră absolută. În lume, literatura despre "metode rapide de modelare" este formată, în principal, din articole disparate apărute în diverse

Scanned with OKEN Scanner

publicaţii recente de specialitate; puţinele cărţi ce abordează subiectul, o fac fără a pătrunde în profunzimea acestuia.

Înţelegerea prezentei lucrări presupune o familiaritate cu noţiunile de bază ale "algebrei matriceale" şi ale "teoriei modelelor dinamice". Străduinţa noastră a fost să facem lucrarea cât mai de sine stătătoare cu putinţă, pentru ca ea să fie utilă unui număr cât mai mare de cititori. Includerea unei biblioteci de programe pentru implementarea principalelor metode rapide de modelare a sistemelor, trebuie înţeleasă în acest sens. În opinia noastră, aceste programe pot fi utilizate în aplicaţii specifice, fără ca acest lucru să necesite studierea profundă a teoriei aferente. Sperăm, deci, că lucrarea se va dovedi utilă şi pentru cercetătorii interesaţi mai mult de aplicaţii practice decât de teorie.

Lucrarea se adresează tuturor celor care operează, într-o formă sau alta, cu modele având un număr relativ mare de parametri: cercetători din domeniile biologiei, ecologiei, econometriei, matematicii, prelucrării semnalelor, automaticii şi electronicii. Scopul principal al lucrării este de a constitui o sursă monografică de referinţă pentru aceşti cercetători. Subcapitolele lucrării pot, de asemenea, să fie utilizate în cursuri avansate pentru studenţii facultăţilor de Automatică, Electronică şi Matematică.

Mulţumim pe această cale Biroului Prezidiului Academiei şi Secţiei de Ştiinţa şi Tehnologia Informaţiei care au sprijinit apariţia cărţii.

Adresăm mulţumiri Institutului de Cercetări în Informatică, şi în mod special directorului acestuia, membru corespondent al Academiei Florin Gheorghe Filip, pentru sponsorizarea tipăririi acestei cărţi.

Autorii

Scanned with OKEN Scanner

Cuprins

Partea întâi. Metode Rapide Recursive în Ordinul Modelului

1. Algoritmul Levinson-Durbin pentru Modele AR Monovariabile 11
- 1.1. Modele AR monovariabile . . . 11
- 1.2. Algoritmul Levinson-Durbin . . . 14
- 1.3. Studiul stabilităţii modelului . . . 17
- 1.4. Utilizări ale algoritmului Levinson- Durbin . . . 22
  - 1.4.1. Factorizarea matricelor inverse de covarianţă . . . 22
  - 1.4.2. Testarea pozitivităţii matricelor Toeplitz . . . 23
  - 1.4.3. Testarea stabilităţii . . . 24

2. Algoritmii Delsarte–Genin şi Schur pentru Modele AR Monovariabile 27
- 2.1. Algoritmul Delsarte–Genin . . . 27
- 2.2. Algoritmul Schur . . . 31

3. Algoritmul Levinson pentru Modele ARMA Monovariabile 39
- 3.1. Modele AR–plus–zgomot şi ARMA . . . 39
- 3.2. Algoritm de tip Levinson . . . 42
- 3.3. Paralelisme între AL şi ALD . . . 46

4. Algoritmul Wittle–Wiggins–Robinson pentru Modele AR Multivariabile 49
- 4.1. Modele AR multivariabile . . . 49
- 4.2. Algoritmul Whittle–Wiggins-Robinson . . . 51
- 4.3. Studiul stabilităţii modelului . . . 56
- 4.4. Utilizări ale algoritmului Wittle–Wiggins–Robinson . . . 59
  - 4.4.1. Factorizarea matricelor inverse de covarianţă . . . 60
  - 4.4.2. Testarea pozitivităţii matricelor Toeplitz de blocuri . . . 61
  - 4.4.3. Testarea stabilităţii . . . 61

5. Algoritmul Levinson pentru Modele Multivariabile 63
- 5.1. Modele de tip sumă de convoluţie . . . 63
- 5.2. Algoritmul Levinson . . . 65

6. Algoritm Rapid pentru Modele Multivariabile de Tip Ecuaţie cu Diferenţe 71
- 6.1. Modele de tip ecuaţie cu diferenţe . . . 71
- 6.2. Studiul stabilităţii modelului . . . 73

Scanned with OKEN Scanner

Partea a doua. Metode Rapide Recursive în Ordinul Modelului şi în Timp

7. Algoritmul Latice pentru Modele AR Monovariabile
7.1. Parametrizarea latice a modelelor AR . . . 77
7.2. Algoritmul "latice a erorilor de predicţie" . . . 77
7.3. Algoritmul latice al predictorilor . . . 86
7.3.1. Deducerea ecuaţiilor de recurenţă în timp . . . 95
7.3.2. Deducerea ecuaţiilor recursive în ordin . . . 95
7.3.3. Variante ale algorimilor latice a erorilor de predicţie şi predictorilor . . . 100
7.3.4. Iniţializarea exactă a algoritmilor latice . . . 105
105

8. Algoritmi Latice Normalizată pentru Modele AR Monovariabile 111
8.1. Algoritmi latice pentru erorile de predicţie normalizate . . . 111
8.2. Algoritmul latice pentru predictorii normalizaţi . . . 117
8.3. Parametrizarea latice normalizată a modelelor AR . . . 122

9. Algoritmul Latice pentru Modele ARMA Monovariabile 125
9.1. Algoritmi latice pentru erorile de predicţie . . . 125
9.2. Trecerea de la parametrii latice la parametrii AR . . . 136

10. Algoritmul Latice pentru Modele Multivariabile de Tip Sumă de Convoluţie 139
10.1. Algoritmul latice pentru erorile de predicţie . . . 139
10.2. Algoritmul "Latice a predictorilor" . . . 154

11. Algoritmul CMMP cu Propagarea Covarianţelor pentru Modele Tip Sumă de Convoluţie 157
11.1. Algoritmul CMMP cu propagarea covarianţelor . . . 157
11.2. Ferestre de date pentru urmărirea parametrilor variabili în timp . . . 161

Partea a treia. Metode Rapide Recursive în Timp

12. Algoritmi Recursivi în Timp pentru Modele AR Monovariabile 169
12.1. Metoda convenţională recursivă în timp a celor mai mici pătrate . . . 169
12.2. Algoritmul Fast Kalman . . . 172
12.3. Algoritmul FAEST . . . 174
12.4. Algoritmul "Filtru Transversal Rapid" . . . 177
12.5. Algoritmul "Filtru Transversal Rapid" normalizat . . . 179

13. Algoritmul "Filtru Transversal Rapid" pentru Modele Multivariabile 185
13.1. Algoritmi recursivi în timp pentru modele multivariabile . . . 185
13.2. Algoritmul "Filtru Transversal Rapid" . . . 185

Anexe. Algoritmi Rapizi Implementaţi în PASCAL şi MATLAB 189

Bibliografie . . . 215
Abstract . . . 222
Contents . . . 223

Scanned with OKEN Scanner

Partea întâi

Metode Rapide Recursive în Ordinul Modelului

Această parte a lucrării prezintă algoritmi rapizi recursivi–în–ordin pentru estimarea parametrilor următoarelor tipuri de modele: modele "autoregresive şi de medie alunecătoare" (ARMA); modele "sumă de convoluţie"; modele "ecuaţie cu diferenţe". Aceşti algoritmi rapizi sunt de tip off-line (ei prelucrează "simultan" întreg şirul de măsurători). Caracteristica lor principală este următoarea: exploatând structura numerică specială a diverselor probleme de estimare considerate, aceşti algoritmi determină parametrii tuturor modelelor cu ordine de la unu până la un ordin maxim preimpus, într-un timp de calcul care este cu două ordine de mărime mai mic decât cel asociat algoritmilor standard de estimare parametrică.

Scanned with OKEN Scanner

Capitolul 1

Algoritmul Levinson-Durbin pentru Modele AR Monovariabile

1.1. Modele AR monovariabile

Fie $\{y(t) \in \mathcal{R}\}_{t=1,2,...}$ un semnal staţionar generat de următoarea ecuaţie:

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = e(t), \tag{1.1}$$

unde $\{a_i\} \in \mathcal{R}$, iar $e(t)$ este un zgomot alb de medie zero şi dispersie σ:

$$\begin{aligned} Ee(t) &= 0, \\ Ee(t)e(s) &= \sigma\delta_{t,s} \end{aligned} \tag{1.2}$$

($\delta_{t,s} = 0$ pentru $t = s$ şi 1 pentru $t \neq s$). Condiţia necesară şi suficientă ca semnalul $\{y(t)\}$ generat de ecuaţia (1.1) să fie staţionar este ca polinomul

$$A(z) = 1 + a_1 z + \ldots + a_n z^n \tag{1.3}$$

să aibă toate zerourile în afara cercului unitar (adică a cercului cu centrul în originea planului complex şi rază egală cu unu)[76]. Utilizând polinomul $A(\cdot)$ şi operatorul de întârziere q^{-1} definit prin

$$q^{-k}y(t) = y(t-k), \quad k = 1, 2, \ldots \tag{1.4}$$

putem scrie ecuaţia (1.1) sub următoarea formă compactă:

$$A(q^{-1})y(t) = e(t). \tag{1.5}$$

Modelul (1.1) (sau (1.5)) pentru $y(t)$ se numeşte *ecuaţie autoregresivă* (AR) din motive destul de transparente: prin intermediul acestei ecuaţii $y(t)$ este exprimat ca o funcţie de valorile sale regresate în timp ($y(t-1), y(t-2)$ etc.). Presupunerea că timpul t ia valori discrete, în ecuaţia (1.1), nu este o restricţie ci mai degrabă o modalitate de simplificare a notaţiei. Intr-adevăr, prin $y(k)$ vom desemna în cele ce urmează valoarea semnalului $y(\cdot)$ la momentul k de eşantionare (cu alte cuvinte $y(k)$ va fi utilizată ca o notaţie simplificată pentru $y(k\Delta)$, unde Δ = perioada de eşantionare).

Modelele AR au aplicaţii multiple în domeniile identificării sistemelor şi prelucrării semnalelor. Ele sunt utilizate ca *predictoare liniare* pentru serii de date temporale sau spaţiale

Scanned with OKEN Scanner

[19],[45],[46]; pentru *analiza spectrală* de mare rezoluţie a semnalelor[16],[17],[54],[55],[96]; în operaţia de *compresiune a datelor*[48]; pentru *sinteza şi recunoaşterea semnalelor vocale* [27],[47], [48],[49] etc. În toate aplicaţiile enumerate, problema de bază este determinarea coeficienţilor $\{a_i\}_{i=1}^n$ şi a dispersiei σ a zgomotului alb din (1.1), prin prelucrarea eşantioanelor disponibile ale semnalului $\{y(t)\}$. Metoda cea mai frecvent utilizată pentru soluţionarea acestei probleme este "metoda celor mai mici pătrate" (CMMP).

În cadrul metodei CMMP, parametrii $\{a_i\}$ sunt determinaţi astfel încât

$$E[y(t) + a_1 y(t-1) + \ldots + a_n y(t-n)]^2 = min, \tag{1.6}$$

unde E este operatorul de mediere statistică. Comparând (1.1) şi (1.6) rezultă că metoda CMMP determină modelul AR prin minimizarea dispersiei "reziduurilor" $\{\epsilon(t)\}$. Pentru a rezolva problema pătratică de minimizare (1.6), să introducem următoarele notaţii:

$$\rho_k = Ey(t)y(t-k) = \rho_{-k}, \tag{1.7}$$

$$R = E\begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} [y(t-1)\ldots y(t-n)] = \begin{bmatrix} \rho_0 & \rho_1 & \cdot & \cdot & \cdot & \rho_{n-1} \\ \rho_1 & \rho_0 & \cdot & \cdot & \cdot & \rho_{n-2} \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ \rho_{n-1} & & \cdot & \cdot & \cdot & \rho_0 \end{bmatrix}, \tag{1.8}$$

$$r = E\begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} y(t) = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \cdot \\ \cdot \\ \cdot \\ \rho_n \end{bmatrix}. \tag{1.9}$$

$$\theta = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}^{\mathrm{T}}. \tag{1.10}$$

Utilizând aceste notaţii putem obţine următoarea formă mai convenabilă a criteriului de minimizat:

$$\begin{aligned} & E[y(t) + a_1 y(t-1) + \ldots + a_n y(t-n)]^2 = \\ = \; & Ey^2(t) + 2\theta^{\mathrm{T}} E\begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} y(t) + \theta^{\mathrm{T}} E\begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} \begin{bmatrix} y(t-1) & \ldots & y(t-n) \end{bmatrix} \theta = \\ = \; & \rho_0 + 2\theta^{\mathrm{T}} r + \theta^{\mathrm{T}} R\theta = \\ = \; & \left(\theta + R^{-1}r\right)^{\mathrm{T}} R\left(\theta + R^{-1}r\right) + \rho_0 - r^{\mathrm{T}} R^{-1} r. \end{aligned} \tag{1.11}$$

Ultima egalitate din (1.11) se bazează pe presupunerea implicită că matricea inversă R^{-1} există, ceea ce este adevărat în condiţii destul de generale asupra semnalului $\{y(t)\}$ [75],[76]. Să observăm acum că matricea R este pozitiv definită. Intr-adevăr, pentru orice vector $\alpha \in \mathcal{R}^{n\times 1}$ avem

$$\alpha^{\mathrm{T}} R\alpha = E\left[\sum_{i=1}^{n} \alpha_i y(t-i)\right]^2 \geq 0 \tag{1.12}$$

Scanned with OKEN Scanner

şi deci R este cel puţin pozitiv semidefinită. Cum $\det R \neq 0$ prin presupunere, urmează că R este o matrice pozitiv definită. Utilizând această observaţie şi ecuaţia (1.11) (în care al doilea termen nu depinde de θ), rezultă imediat că soluţia θ a problemei de minimizare (1.6) este următoarea:

$$\theta = -R^{-1}r. \tag{1.13}$$

Comparând (1.1) şi (1.11) observăm că valoarea criteriului din membrul stâng al ecuaţiei (1.11), în punctul de minim, este egală cu dispersia σ a zgomotului $\{e(t)\}$. Astfel,

$$\sigma = \rho_0 - r^{\mathrm{T}}R^{-1}r = \rho_0 + r^{\mathrm{T}}\theta, \tag{1.14}$$

unde prima egalitate urmează din (1.11), iar a doua din (1.13).

În practică ecuaţiile (1.13) şi (1.14) nu pot fi utilizate direct pentru determinarea parametrilor $\{\theta, \sigma\}$ ai modelului AR, pentru simplul motiv că secvenţa de covarianţe teoretice $\{\rho_k\}_{k=0}^{n}$, care apar în R şi r, nu este disponibilă.

Soluţia practică cea mai naturală a acestei dileme constă în înlocuirea covarianţelor teoretice din R şi r cu aşa numitele "covarianţe de eşantion":

$$\rho_k = \frac{1}{N}\sum_{t=k+1}^{N} y(t)y(t-k), \tag{1.15}$$

unde N este numărul de măsurători disponibile. Pentru a nu complica notaţia, nu vom introduce o notaţie adiţională pentru covarianţele de eşantion (de exemplu, $\{\hat{\rho}_k\}$) şi nici pentru matricea (1.8) şi vectorul (1.9) construite din aceste covarianţe.

Să rezumăm discuţia anterioară: estimaţiile CMMP ale parametrilor $\{\theta$ şi $\sigma\}$ ai modelului AR (1.1) sunt date de ecuaţiile (1.13) şi (1.14), unde R şi r sunt construite din covarianţele de eşantion (1.15). Matricea R în (1.13) are o structură foarte specială. Mai exact, elementul i, j al acestei matrice

$$R_{i,j} = \rho_{|i-j|} \tag{1.16}$$

nu depinde de i şi j, ci numai de modulul diferenţei $|i-j|$. O astfel de matrice se numeşte *Toeplitz simetrică*. Dacă această structură particulară a lui R este ignorată şi, deci, sistemul de ecuaţii liniare (1.13) este rezolvat utilizând un algoritm standard, atunci numărul de operaţii aritmetice necesare calculului lui θ (şi σ) ar fi de ordinul lui n^3: $\mathcal{O}(n^3)$[100]. În majoritatea aplicaţiilor modelului AR (1.1), ordinul n al acestuia este necunoscut. Procedura uzuală pentru determinarea lui n constă în determinarea şi testarea tuturor modelelor de ordine $1, 2, \ldots, n_{max}$ (unde n_{max} este o valoare presupusă cunoscută, care este astfel încât $n \leq n_{max}$). Numărul total de operaţii necesare determinării întregului set de modele cu ordine $1, 2, \ldots, n_{max}$ va fi $\mathcal{O}(n_{max}^4)$. Dacă structura Toeplitz a lui R este exploatată, atunci acest număr poate fi micşorat cu două ordine de mărime: de la $\mathcal{O}(n_{max}^4)$ la $\mathcal{O}(n_{max}^2)$! Această scădere considerabilă a numărului de operaţii este extrem de importantă pentru aplicaţiile modelelor AR care consideră modele de ordine relativ mari. De exemplu, determinarea frecvenţelor semnalelor sinusoidale din măsurători perturbate poate necesita utilizarea unor modele AR de ordine $n \sim 50$ sau chiar $n \sim 100$ [57],[82].

În paragraful următor vom descrie în detaliu modul în care structura particulară a ecuaţiilor (1.13),(1.14) poate fi exploatată pentru a obţine algoritmul rapid Levinson-Durbin de rezolvare a acestor ecuaţii.

Scanned with OKEN Scanner

1.2. Algoritmul Levinson-Durbin

Să ataşăm cantităţilor ce apar în (1.13) şi (1.14) un index n, pentru a sublinia faptul că ele corespund modelului de ordin n. Astfel, (1.13) devine

$$R_n\theta_n = -r_n, \tag{1.17}$$

iar (1.14)

$$\sigma_n = \rho_0 + r_n^{\mathrm{T}}\theta_n. \tag{1.18}$$

Ecuaţiile (1.17) şi (1.18) pot fi combinate pentru a obţine un singur sistem liniar cu $(n+1)$ necunoscute:

$$R_{n+1}\begin{bmatrix} 1 \\ \theta_n \end{bmatrix} = \begin{bmatrix} \sigma_n \\ 0 \end{bmatrix}. \tag{1.19}$$

Pentru a verfica (1.19) este suficient să notăm următoarea partiţionare a matricei R_{n+1}:

$$R_{n+1} \triangleq \begin{bmatrix} \rho_0 & \rho_1 & \cdot & \cdot & \cdot & \rho_n \\ \rho_1 & \rho_0 & \cdot & \cdot & \cdot & \rho_{n-1} \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ \rho_n & & \cdot & & & \rho_0 \end{bmatrix} = \begin{bmatrix} \rho_0 & r_n^{\mathrm{T}} \\ r_n & R_n \end{bmatrix}. \tag{1.20}$$

Vom prezenta în continuare ceea ce considerăm a fi cea mai simplă deducere posibilă a algoritmului Levinson-Durbin (ALD). Alte deduceri pot fi găsite în [2],[3],[14],[23]. Introducem notaţiile:

$$\alpha_n = \rho_{n+1} + \theta_n^{\mathrm{T}}\tilde{r}_n, \tag{1.21}$$

$$k_{n+1} = -\frac{\alpha_n}{\sigma_n}. \tag{1.22}$$

Simbolul " ˜ " ataşat unui vector îi rearanjează elementele în ordine inversă. Astfel, dacă $x = [x_1 \ldots x_n]^{\mathrm{T}}$ atunci $\tilde{x} = [x_n \ldots x_1]^{\mathrm{T}}$. Referitor la operaţia de permutare implicată de " ˜ ", matricele Toeplitz au următoarea proprietate simplă dar esenţială pentru cele ce urmează.

Lema 1.2.1. *Fie R o matrice Toeplitz simetrică, şi fie*

$$y = Rx. \tag{1.23}$$

Atunci

$$\tilde{y} = R\tilde{x}. \tag{1.24}$$

Demonstraţie: Din (1.23) urmează că elementul i al vectorului $\tilde{y}$ are următoarea expresie

$$\begin{aligned} \tilde{y}_i &= y_{n-i+1} = \sum_{k=1}^{n} R_{n-i+1,k}x_k = \\ &= \sum_{k=1}^{n} \rho_{|n-i+1-k|}x_k = \sum_{p=1}^{n} \rho_{|p-i|}x_{n-p+1} = \\ &= \sum_{p=1}^{n} R_{p,i}\tilde{x}_p = (R\tilde{x})_i \end{aligned} \tag{1.25}$$

Scanned with OKEN Scanner

şi, deci, egalitatea (1.24) este demonstrată.
□

Suntem acum în măsură să prezentăm o deducere foarte simplă a ALD.

Teorema 1.2.1. *Soluţiile de ordinul n şi $n+1$ ale sistemului de ecuaţii liniare* (1.19) *sunt legate prin următoarele relaţii:*

$$\theta_{n+1} = \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} + k_{n+1} \begin{bmatrix} \tilde{\theta}_n \\ 1 \end{bmatrix}, \tag{1.26}$$

$$\sigma_{n+1} = \sigma_n(1 - k_{n+1}^2), \tag{1.27}$$

unde k_{n+1} este definit în (1.22).

Demonstraţie: Să observăm utilizând (1.19) că,

$$R_{n+2} \begin{bmatrix} 1 \\ \theta_n \\ 0 \end{bmatrix} = \left[\begin{array}{c|c} R_{n+1} & \begin{matrix} \rho_{n+1} \\ \tilde{r}_n^- \end{matrix} \\ \hline \begin{matrix} \rho_{n+1} & \tilde{r}_n^T \end{matrix} & \rho_0 \end{array}\right] \begin{bmatrix} 1 \\ \theta_n \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_n \\ 0 \\ \alpha_n \end{bmatrix}. \tag{1.28}$$

Urmează atunci imediat din Lema 1.2.1. că θ_{n+1} şi σ_{n+1} definite în (1.26) (1.27) satisfac ecuaţia de mai jos:

$$\begin{aligned} R_{n+2} \begin{bmatrix} 1 \\ \theta_{n+1} \end{bmatrix} &= R_{n+2} \left\{ \begin{bmatrix} 1 \\ \theta_n \\ 0 \end{bmatrix} + k_{n+1} \begin{bmatrix} 0 \\ \tilde{\theta}_n \\ 1 \end{bmatrix} \right\} = \\ &= \begin{bmatrix} \sigma_n \\ 0 \\ \alpha_n \end{bmatrix} + k_{n+1} \begin{bmatrix} \alpha_n \\ 0 \\ \sigma_n \end{bmatrix} = \\ &= \begin{bmatrix} \sigma_n + k_{n+1}\alpha_n \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_n(1 - k_{n+1}^2) \\ 0 \end{bmatrix}, \end{aligned} \tag{1.29}$$

unde am utilizat, de asemenea, faptul că $\alpha_n + k_{n+1}\sigma_n = 0$ (vezi (1.22)). Comparând (1.29) cu (1.19) şi făcând uz de faptul că soluţia sistemului (1.19) pentru un n dat este unică prin presupunere, rezultă imediat că soluţia de ordin $n+1$ este într-adevăr dată de relaţiile (1.26) şi (1.27).
□

Relaţiile (1.26) şi (1.27) constituie ecuaţiile de bază ale ALD. De fapt, pentru obţinerea ALD singurul lucru necesar este completarea ecuaţiilor recursive-în-ordin (1.26), (1.27) cu expresii pentru θ_1 şi σ_1. Expresiile pentru θ_1 şi σ_1 sunt, totuşi, uşor de obţinut din relaţiile de definiţie (1.17) şi (1.18):

$$\theta_1 = -\frac{\rho_1}{\rho_0}, \tag{1.30}$$

$$\sigma_1 = \rho_0 - \frac{\rho_1^2}{\rho_0}. \tag{1.31}$$

Scanned with OKEN Scanner

Ecuaţie	Număr operaţii [flop]
Iniţializare:	
$\theta_1 = -\frac{\rho_1}{\rho_0} = k_1$	$\mathcal{O}(1)$
$\sigma_1 = \rho_0 - \frac{\rho_1^2}{\rho_0}$	$\mathcal{O}(1)$
Pentru $n = 1, \ldots, n_{max}$ calculează:	
$k_{n+1} = -(\rho_{n+1} + \theta_n^{\mathrm{T}} \tilde{r}_n)/\sigma_n$	$\sim n$
$\sigma_{n+1} = \sigma_n(1 - k_{n+1}^2)$	$\mathcal{O}(1)$
$\theta_{n+1} = \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} + k_{n+1} \begin{bmatrix} \tilde{\theta}_n \\ 1 \end{bmatrix}$	$\sim n$

Tabelul 1.1. Algoritmul Levinson-Durbin.

Ecuaţiile (1.26),(1.27),(1.30) şi (1.31), care împreună formează ALD, sunt prezentate sintetic în Tabelul 1.1. împreună cu numărul de operaţii necesare pentru implementare. Acest număr de operaţii este exprimat în "flopi", unde

$$1 \text{ flop} = \text{o adunare şi o înmulţire (sau împărţire) în virgulă mobilă.} \tag{1.32}$$

Observăm din Tabelul 1.1. că numărul de "flopi" necesari algoritmului Levinson-Durbin pentru a trece de la modelul de ordin n la cel de ordin $n+1$ este $\sim 2n$. Astfel, numărul total de operaţii aritmetice efectuate de ALD pentru a calcula toate modelele cu ordine $n = 1, 2, \ldots, n_{max}$ este

$$2 \sum_{n=1}^{n_{max}} n = n_{max}^2 + n_{max}[flop]. \tag{1.33}$$

ALD este, deci, cu două ordine de mărime mai rapid decât un algoritm standard (ca, de exemplu, cel bazat pe eliminare gaussiană) aplicat sistemelor de ecuaţii liniare (1.19). În plus, ALD este extrem de simplu de codificat într-un limbaj de înalt nivel (vezi Tabelul 1.1.). În fine, ALD are şi proprietatea importantă de a fi numeric stabil[7]. În legătură cu această ultimă proprietate, să facem următoarea observaţie. Datorită presupunerilor făcute, matricea $R_{n_{max}}$ este pozitiv definită. Acest fapt şi unele proprietăţi binecunoscute ale matricelor pozitiv definite partiţionate asigură că

$$\sigma_n > 0, \qquad n = 1, 2, \ldots, n_{max}. \tag{1.34}$$

Totuşi, dacă matricea $R_{n_{max}}$ are valori proprii aproape nule, atunci pot exista $\{\sigma_n\}$ cu valori apropiate de zero. În astfel de cazuri, rezultatele obţinute cu ALD pot fi afectate de erori numerice ne–neglijabile. Această comportare se datoreşte, totuşi, nu unor deficienţe ale ALD, ci faptului că problema rezolvării sistemului (1.19) cu R_n cvasi–singular este prost condiţionată numeric.

Scanned with OKEN Scanner

1.3. Studiul stabilităţii modelului

Dacă semnalul $y(t)$ este într-adevăr generat de o ecuaţie AR de ordin n (vezi (1.1)), atunci se poate arăta uşor că elementele lui θ_n tind la valorile adevărate ale coeficienţilor $\{a_i\}$ ai ecuaţiei AR, când N creşte. Cu alte cuvinte, ALD furnizează estimaţii *consistente* ale parametrilor modelelor AR.

În aplicaţii practice, totuşi, $y(t)$ nu va fi niciodată generat de o ecuaţie AR pură. Ca atare nu mai putem vorbi de convergenţa estimaţiilor la valorile adevărate ale parametrilor pentru simplul fapt că astfel de valori nu mai există în astfel de cazuri. Putem totuşi pune următoarea întrebare:

> Este modelul AR determinat cu ALD stabil în condiţii generale (adică pentru semnale $y(t)$ staţionare generale)? (1.35)

Răspunsul la această întrebare este important pentru majoritatea aplicaţiilor modelelor AR, care se bazează pe ipoteza stabilităţii modelului. Teorema următoare conţine răspunsul la întrebarea (1.35).

Teorema 1.3.1. *Următoarele afirmaţii sunt echivalente:*

(i) $|k_p| < 1 \quad (p = 1, \ldots, n)$ *şi* $\rho_0 > 0$;

(ii) $R_{n+1} > 0$ *(adică matricea* R_{n+1} *este pozitiv definită);*

(iii) Vectorul θ_n *defineşte un model AR stabil, adică* $A_n(z) \triangleq 1 + [z \ldots z^n]\theta_n \neq 0$ *pentru* $|z| \leq 1$ *şi* $\rho_0 > 0$.

Demonstraţie: Să considerăm mai întâi echivalenţa (i) $\Leftrightarrow$ (ii). Calcule simple arată că (vezi(1.19))

$$\begin{bmatrix} 1 & \theta_n^T \\ 0 & I \end{bmatrix} R_{n+1} \begin{bmatrix} 1 & 0 \\ \theta_n & I \end{bmatrix} = \begin{bmatrix} \sigma_n & 0 \\ 0 & R_n \end{bmatrix} \tag{1.36}$$

şi deci

$$\det R_{n+1} = \sigma_n \det R_n. \tag{1.37}$$

Matricea R_{n+1} este pozitiv definită dacă şi numai dacă

$$\det R_1 = \rho_0 > 0; \quad \det R_p > 0; \quad p = 2, \ldots, n+1. \tag{1.38}$$

Totuşi, urmează din (1.37) că (1.38) este echivalentă cu următoarea condiţie

$$\rho_0 > 0; \quad \sigma_p > 0; \quad p = 1, \ldots, n, \tag{1.39}$$

care la rândul ei este echivalentă cu (i). Astfel echivalenţa (i) $\Leftrightarrow$ (ii) este demonstrată.

Să considerăm în continuare echivalenţa (i) $\Leftrightarrow$ (iii). Pentru a demonstra această echivalenţă facem uz de următoarea teoremă a lui Rouché:

> Dacă $f(z)$ şi $g(z)$ sunt două funcţii analitice în interiorul şi pe o curbă închisă $\mathcal{C}$, iar $|g(t)| < |f(t)|$ pentru $z \in \mathcal{C}$, atunci $f(z)$ şi $f(z) + g(z)$ au acelaşi număr de zerouri în interiorul lui $\mathcal{C}$. (1.40)

Scanned with OKEN Scanner

Ecuaţia iterativă (1.26) poate fi scrisă sub următoarea formă polinomială:

$$A_{p+1}(z) = A_p(z) + k_{p+1} z^{p+1} A_p(z^{-1}), \tag{1.41}$$

unde $A_p(z)$ şi $A_{p+1}(z)$ sunt definite similar cu $A_n(z)$ din enunţul teoremei. Să presupunem că inegalităţile (i) sunt satisfăcute. Atunci, pentru $|z| = 1$,

$$|A_p(z)| = |z^{p+1} A_p(z^{-1})| > |k_{p+1} z^{p+1} A_p(z^{-1})| \tag{1.42}$$

şi deci, în conformitate cu (1.40), $A_{p+1}(z)$ şi $A_p(z)$ au acelaşi număr de zerouri în interiorul cercului unitar. Deoarece

$$A_1(z) = 1 + \theta_1 z = 1 + k_1 z \tag{1.43}$$

nu are nici un zero în interiorul cercului unitar cînd condiţia (i) este satisfăcută, rezultă că $A_p(z)(p = 1, \ldots, n)$ au toate zerourile strict în afara discului unitar. Deci (i) $\Leftrightarrow$ (iii).

În fine, să considerăm că (iii) este satisfăcut.

Cum coeficientul lui z^n în $A_n(z)$ este k_n (vezi (1.26)), urmează că $|k_n| < 1$. De asemenea urmează că

$$A_{n-1}(z) \neq 0 \text{ pentru } |z| \leq 1. \tag{1.44}$$

Intr-adevăr, dacă $A_{n-1}(z)$ ar avea zerouri în interiorul cercului unitar, atunci din $\{(1.41), |k_n| < 1$ şi $(1.40)\}$ ar rezulta că $A_n(z)$ ar trebui să aibă unele zerouri în interiorul cercului unitar, ceea ce ar contrazice (iii). La rândul ei, ecuaţia (1.44) implică $|k_{n-1}| < 1$ şi deci putem continua în maniera anterioară pentru a conchide că (i) este valid. Astfel teorema este demonstrată.
□

Teorema 1.3.1. are un număr de consecinţe importante pentru utilitatea ALD. Astfel, echivalenţele (i) $\Leftrightarrow$ (ii) şi (i) $\Leftrightarrow$ (iii) pot fi folosite pentru conceperea unor proceduri utilizabile pentru a testa dacă o matrice Toeplitz dată este pozitiv definită şi, respectiv, dacă un polinom dat este stabil. Aceste utilizări ale ALD sunt discutate în paragraful următor. În prezentul paragraf vom utiliza echivalenţa (ii) $\Leftrightarrow$ (iii) pentru a studia stabilitatea modelului AR determinat cu ALD.

Matricea de covarianţă R_{n+1} este teoretic pozitiv definită. Totuşi rămâne ea pozitiv definită în cazul, relevant practic, când covarianţele teoretice sunt înlocuite prin covarianţe de eşantion? În cazul în care covarianţele de eşantion sunt determinate cu formula (1.15), matricea R_{n+1} corespunzătoare este pozitiv definită în condiţii generale asupra semnalului $y(t)$. Intr-adevăr, în acest caz, R_{n+1} poate fi scris în următoarea formă

$$R_{n+1} = \frac{1}{N} \begin{bmatrix} y(1) & \cdot & \cdot & \cdot & y(N) & 0 & \cdot & \cdot & 0 \\ 0 & y(1) & & & & y(N) & 0 & & 0 \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & & & & \cdot & 0 \\ 0 & \cdot & & 0 & y(1) & & & & y(N) \end{bmatrix} \begin{bmatrix} y(1) & 0 & & 0 \\ \cdot & y(1) & & 0 \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & 0 \\ \cdot & & & y(1) \\ y(N) & & & \cdot \\ 0 & y(N) & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ 0 & \cdot & 0 & y(N) \end{bmatrix} \tag{1.45}$$

şi deci este o matrice pozitiv semidefinită pentru orice secvenţă $\{y(1), \ldots, y(N)\}$ a semnalului $y(t)$ şi pozitiv definită pentru o clasă largă de secvenţe. Urmează atunci din echivalenţa (ii)

Scanned with OKEN Scanner

$\Leftrightarrow$ (iii) a Teoremei 1.3.1. că *modelul AR determinat cu ALD prin prelucrarea covarianţelor de eşantion (1.15) este stabil.* Să remarcăm că alte covarianţe de eşantion, cum ar fi aşa numitele " covarianţe *nedeviate* de eşantion",

$$\rho_k = \frac{1}{N-k} \sum_{t=k+1}^{N} y(t)y(t-k) \tag{1.46}$$

nu posedă în mod necesar proprietatea (ii) şi deci *nu* garantează stabilitatea modelului AR determinat prin prelucrarea lor cu ALD.

O întrebare interesantă legată de discuţia anterioară este următoarea: ce se întâmplă dacă matricea R_{n+1} nu este strict pozitiv definită ci numai pozitiv semidefinită (prin urmare $\det R_{n+1} = 0$)? După cum rezultă din (1.45), o astfel de situaţie ar putea apare pentru unele secvenţe $\{y(1), \ldots, y(N)\}$ "degenerate". Răspunsul la întrebarea anterioară este furnizat în cele ce urmează pentru cazul în care $|k_p| < 1$ $(p = 1, \ldots, n-1)$. Această condiţie nu constituie efectiv o restricţie, pentru că dacă $|k_p| = 1$ pentru $p \leq n-1$, atunci ALD trebuie stopat înainte de terminare (vezi Tabelul 1.1.) şi deci polinomul $A_n(z)$ nu mai poate fi calculat.

Lema 1.3.1. *Să presupunem $|k_p| < 1$ $(p = 1, \ldots, n-1)$ şi că $\rho_0 > 0$. Atunci următoarele afirmaţii sunt echivalente:*

(i) $|k_n| = 1$.

(ii) $\det R_{n+1} = 0$.

(iii) $A_n(z) = 0 \quad \Rightarrow |z| = 1$.

Demonstraţie: Prin iterarea lui (1.37) se obţine:

$$\det R_{n+1} = \rho_0 \prod_{k=1}^{n} \sigma_k. \tag{1.47}$$

Echivalenţa (i) $\Leftrightarrow$ (ii) urmează imediat din (1.27) şi (1.47). Implicaţia (iii) $\Leftrightarrow$ (i) este, de asemenea, imediată deoarece k_n este coeficientul lui z^n în $A_n(z)$. Rămâne de demonstrat că (i) $\Rightarrow$ (iii). Fie k_n^ϵ obţinut prin perturbarea uşoară a lui k_n ($|k_n| = 1$), şi fie

$$A_n^\epsilon(z) \triangleq A_{n-1}(z) + k_n^\epsilon z^n A_{n-1}(z^{-1}). \tag{1.48}$$

Comparând (1.41) şi (1.48) se observă uşor că $A_n^\epsilon(z)$ se poate obţine prin perturbarea coeficienţilor polinomului $A_n(z)$:

$$A_n^\epsilon(z) = A_n(z) + (k_n^\epsilon - k_n) z^n A_{n-1}(z^{-1}). \tag{1.49}$$

Pentru $|k_n^\epsilon| < 1$ urmează din Teorema 1.3.1. că $A_n^\epsilon(z)$ are toate zerourile în afara cercului unitar. În schimb, pentru $|k_n^\epsilon| > 1$, deoarece

$$|A_{n-1}(z)| < |k_n^\epsilon||z^n A_{n-1}(z)| \text{ pentru } |z| = 1, \tag{1.50}$$

urmează din (1.40) că $A_n^\epsilon(z)$ şi $z^n A_{n-1}(z^{-1})$ au acelaşi număr de zerouri în discul unitar. Astfel, pentru $|k_n^\epsilon| > 1$, $A_n^\epsilon(z)$ are toate zerourile în interiorul discului unitar.

Totuşi zerourile unui polinom sunt funcţii continue de coeficienţii acestuia. Prin urmare, zerourile unui polinom uşor perturbat nu pot fi fie toate în afara discului unitar, fie toate în

Scanned with OKEN Scanner

disc, în funcţie de perturbaţie, în afară de cazul în care polinomul neperturbat are toate zerourile pe cercul unitar. Cu această observaţie, lema este demonstrată.
□

Lema 1.3.1. constituie o extensie interesantă a Teoremei 1.2.1.. Ea arată că pentru matrice R_{n+1} cvasi-singulare, modelul AR determinat cu ALD va fi aproape de limita de stabilitate (rezultatul lemei este neaşteptat de net: la limită, când matricea R_{n+1} este exact singulară, *toate* zerourile lui $A_n(z)$ – şi nu numai o parte din ele –migrează pe cercul unitar).

În unele aplicaţii este util să ştim nu numai că modelul AR este stabil, ci şi cât de aproape de limita de stabilitate este acest model (din discuţia anterioară a rezultat că el poate fi "stabil la limită" când matricea R_{n+1} este cvasi-singulară – o astfel de situaţie poate crea dificultăţi în unele aplicaţii, în special dacă nu este cunoscută). Calea cea mai directă pentru a studia "*gradul de stabilitate*" al modelului AR determinat cu ALD constă în calcularea zerourilor polinomului $A_n(t)$. O posibilitate mai simplă ar fi determinarea unor limite superioare pe modulurile acestor zerouri, limite care să poată fi uşor calculate din elementele lui R_{n+1}. Teorema următoare deduce o astfel de limită.

Teorema 1.3.2. *Să introducem următoarele notaţii ($\{a_i\}$ sunt elementele vectorului θ_n):*

$$F = \begin{bmatrix} 1 & 0 & . & . & 0 \\ a_1 & 1 & 0 & & . \\ . & . & . & . & . \\ . & & a_1 & 1 & 0 \\ a_{n-1} & . & . & a_1 & 1 \end{bmatrix}, \tag{1.51}$$

$$G = \begin{bmatrix} a_n & a_{n-1} & . & . & . & a_1 \\ 0 & a_n & a_{n-1} & . & . & a_2 \\ . & . & . & . & & \\ . & & & 0 & a_n & a_{n-1} \\ 0 & 0 & & . & 0 & a_n \end{bmatrix}, \tag{1.52}$$

$$\overline{R}_n = F^{\mathrm{T}} R_n F, \tag{1.53}$$

$$\underline{R}_n = G^{\mathrm{T}} R_n G. \tag{1.54}$$

Fie λ un zero generic al polinomului $z^n A_n(z^{-1})$ şi să presupunem că $R_{n+1} > 0$. Atunci:

$$|\lambda| \leq \min \left[(1 - \frac{\sigma_n}{\operatorname{tr} \overline{R}_n})^{\frac{1}{2n}}, \frac{1}{(1 + \frac{\sigma_n}{\operatorname{tr} \underline{R}_n})^{\frac{1}{2n}}} \right]. \tag{1.55}$$

Demonstraţie: Zerourile $\{\lambda\}$ ale lui $z^n A_n(z^{-1})$ sunt egale cu valorile proprii ale următoarei matrice companion

$$A_c = \begin{bmatrix} -a_1 & 1 & 0 & . & . & 0 \\ -a_2 & 0 & 1 & 0 & . & 0 \\ . & & . & . & . & \\ & & & . & . & 0 \\ -a_{n-1} & 0 & & 0 & 1 & \\ -a_n & 0 & . & . & . & 0 \end{bmatrix}. \tag{1.56}$$

Fie $u = [u_1 \dots u_n]^{\mathrm{T}} \neq 0$ vectorul propriu al lui A_c, asociat cu λ. Astfel,

$$A_c u = \lambda u. \tag{1.57}$$

Scanned with OKEN Scanner

Este uşor de văzut din (1.56) şi (1.57) că $u \neq 0$ implică $u_1 \neq 0$. Putem, deci, să-l normalizăm pe u asfel încât $u_1 = 1$. Inserând $u_1 = 1$ în (1.57), obţinem următoarele ecuaţii:

$$\begin{aligned} u_1 &= 1 \\ u_2 - a_1 &= \lambda u_1 \\ &\cdot \\ &\cdot \\ &\cdot \\ u_n - a_{n-1} &= \lambda u_{n-1} \end{aligned} \tag{1.58}$$

$$-a_n = \lambda u_n, \tag{1.59}$$

care pot fi scrise într-o formă mai compactă:

$$\begin{bmatrix} u \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \theta_n \end{bmatrix} + \lambda \begin{bmatrix} 0 \\ u \end{bmatrix}. \tag{1.60}$$

Din (1.19),(1.60) şi structura specială a lui R_{n+1}, rezultă că (simbolul "$\star$" denotă operaţia de transpunere şi complex conjugare):

$$\begin{aligned} u^\star R_n u &= \begin{bmatrix} u^\star & 0 \end{bmatrix} R_{n+1} \begin{bmatrix} u \\ 0 \end{bmatrix} = \\ &= \left\{ \begin{bmatrix} 1 & \theta_n^T \end{bmatrix} + \lambda^\star \begin{bmatrix} 0 & u^\star \end{bmatrix} \right\} R_{n+1} \left\{ \begin{bmatrix} 1 \\ \theta_n \end{bmatrix} + \lambda \begin{bmatrix} 0 \\ u \end{bmatrix} \right\} = \\ &= \sigma_n + |\lambda|^2 u^\star R_n u. \end{aligned} \tag{1.61}$$

Astfel,

$$|\lambda|^2 = 1 - \frac{\sigma_n}{u^\star R_n u}. \tag{1.62}$$

Din ecuaţille (1.59) obţinem relativ uşor următoarele două expresii pentru u:

$$u = F \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ \lambda^{n-1} \end{bmatrix} \triangleq F\rho \tag{1.63}$$

şi

$$u = -\frac{1}{\lambda^n} G\rho \tag{1.64}$$

(în (1.64) s-a făcut presupunerea implicită că $\lambda \neq 0$). Utilizând (1.63),(1.64) şi o proprietate simplă a funcţiei "trace" obţinem:

$$u^\star R_n u = \rho^\star \overline{R}_n \rho \leq \rho^\star \rho \mathrm{tr} \overline{R}_n = \frac{1 - |\lambda|^{2n}}{1 - |\lambda|^2} \mathrm{tr} \overline{R}_n \tag{1.65}$$

şi respectiv,

$$u^\star R_n u = \frac{\rho^\star \tilde{R}_n \rho}{|\lambda|^{2n}} \leq \frac{\rho^\star \rho \mathrm{tr} \tilde{R}_n}{|\lambda|^{2n}} = \frac{1}{|\lambda|^{2n}} \cdot \frac{1 - |\lambda|^{2n}}{1 - |\lambda|^2} \mathrm{tr} \tilde{R}_n. \tag{1.66}$$

Scanned with OKEN Scanner

Inegalitatea (1.55) urmează imediat din (1.62),(1.65) şi (1.66).
□

Pentru valori mari ale lui n, membrul drept al inegalităţii (1.55) este apropiat de unu. În astfel de cazuri informaţia adiţională, furnizată de (1.55), privind gradul de stabilitate al modelului AR determinat cu ALD, este insignifiantă. Totuşi, pentru valori rezonabile ale lui n, limita superioară (1.55) pe polii modelului AR poate fi suficient de informativă; în plus ea este şi simplu de calculat. Aceste ultime afirmaţii sunt ilustrate în continuare prin intermediul a două exemple numerice.

Fie $\rho_0 = 1,14$, $\rho_1 = 0,28$ şi $\rho_2 = -0,20$. Atunci, după unele calcule simple,

$$z^2 A_2(z^{-1}) = z^2 - 0,31z + 0,25$$

cu zerouri $\lambda_{1,2} = 0,5e^{\pm 0,4\pi j}$; astfel $|\lambda| = 0,5$, iar din (1.55) obţinem:

$$|\lambda| \leq \min(0,86, 0,64) = 0,64.$$

În fine , să presupunem că $\rho_0 = 3,21$, $\rho_1 = 0,99$ şi $\rho_2 = -2,1$, pentru care:

$$z^2 A_2(z^{-1}) = z^2 - 0,56z + 0,81.$$

Zerourile acestui polinom sunt $\lambda_{1,2} = 0,9e^{\pm 0,4\pi j}$; astfel $|\lambda| = 0,9$. Din (1.55) obţinem:

$$|\lambda| \leq \min(0,96, 0,95) = 0,95.$$

Ambele limite superioare calculate mai sus sunt rezonabil de precise.

1.4. Utilizări ale algoritmului Levinson– Durbin

Aşa cum a fost prezentat până acum, ALD este o *procedură rapidă pentru estimarea parametrilor modelelor AR* prin prelucrarea covarianţelor de eşantion ale datelor. Totuşi, aceasta nu este singura utilizare posibilă a ALD. Algoritmul rapid Levinson–Durbin mai are un număr mare de alte utilizări interesante care vor fi descrise în această secţiune.

1.4.1. Factorizarea matricelor inverse de covarianţă

Intr-o serie de aplicaţii, cum ar fi determinarea parametrilor semnalelor sinusoidale utilizând metoda Yule –Walker optimal ponderată[84], apare următoarea problemă: pentru o matrice de covarianţă dată, R_{n+1}, se doreşte calculul factorizării UDU^{T} a lui R_{n+1}^{-1} (adică descompunerea matricei inverse R_{n+1}^{-1} în produsul a 3 matrice: o matrice superior triunghiulară U, o matrice diagonală D şi matricea inferior triunghiulară U^{T}).

Următoarea teoremă demonstrează că ALD calculează, în mod implicit, descompunerea UDU^{T} a matricei inverse de covarianţă R_{n+1}^{-1} (să notăm în trecere că, deşi R_{n+1} este o matrice Toeplitz, matricea inversă R_{n+1}^{-1} nu mai are în general această structură specială).

Scanned with OKEN Scanner

Teorema 1.4.1. *Parametrii* $\{\theta_k, \sigma_k\}_{k=1}^n$ *calculaţi de ALD satisfac următoarea ecuaţie matriceală:*

$$\underbrace{\begin{bmatrix} 1 & & \theta_n^{\mathrm{T}} \\ 0 & 1 & \theta_{n-1}^{\mathrm{T}} \\ & \ddots & \ddots \\ 0 & 0 & 1 \end{bmatrix}}_{U_{n+1}^{\mathrm{T}}} \underbrace{\begin{bmatrix} \rho_0 & \rho_1 & \cdots & \rho_n \\ \rho_1 & \rho_0 & \cdots & \rho_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_n & & \cdots & \rho_0 \end{bmatrix}}_{R_{n+1}} \underbrace{\begin{bmatrix} 1 & 0 & & 0 \\ & \ddots & \ddots & \\ \theta_n & & \ddots & \\ & & 1 & 0 \\ & & \theta_1 & 1 \end{bmatrix}}_{U_{n+1}} = \underbrace{\begin{bmatrix} \sigma_n & 0 & & 0 \\ 0 & \sigma_{n-1} & & \\ & \ddots & \ddots & \\ & & 0 & \sigma_1 & 0 \\ 0 & & & 0 & \rho_0 \end{bmatrix}}_{D_{n+1}} \tag{1.67}$$

Demonstraţie: Ecuaţia (1.67), pentru $n = 1$,

$$\begin{bmatrix} 1 & \theta_1 \\ 0 & 1 \end{bmatrix} R_2 \begin{bmatrix} 1 & 0 \\ \theta_1 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \rho_0 \end{bmatrix} \tag{1.68}$$

urmează direct din (1.19).

Să presupunem acum că ecuaţia (1.67) este validă pentru $n+1 = k$, adică

$$U_k^{\mathrm{T}} R_k U_k = D_k \tag{1.69}$$

şi să arătăm că ea va fi atunci satisfăcută şi pentru $n+1 = k+1$. De fapt, acest lucru urmează după câteva calcule fără complicaţii:

$$\begin{aligned} U_{k+1}^{\mathrm{T}} R_{k+1} U_{k+1} &= \begin{bmatrix} 1 & \theta_k^{\mathrm{T}} \\ 0 & U_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \rho_0 & r_k^{\mathrm{T}} \\ r_k & R_k \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \theta_k & U_k \end{bmatrix} = \\ &= \begin{bmatrix} \rho_0 + \theta_k^{\mathrm{T}} r_k & r_k^{\mathrm{T}} + \theta_k^{\mathrm{T}} R_k \\ U_k^{\mathrm{T}} r_k & U_k^{\mathrm{T}} R_k \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \theta_k & U_k \end{bmatrix} = \\ &= \begin{bmatrix} \sigma_k & 0 \\ U_k^{\mathrm{T}} r_k & U_k^{\mathrm{T}} R_k \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \theta_k & U_k \end{bmatrix} = \\ &= \begin{bmatrix} \sigma_k & 0 \\ U_k^{\mathrm{T}}(r_k + R_k \theta_k) & U_k^{\mathrm{T}} R_k U_k \end{bmatrix} = \begin{bmatrix} \sigma_k & 0 \\ 0 & D_k \end{bmatrix} = D_{k+1}. \end{aligned} \tag{1.70}$$

Raţionamentul inductiv de mai sus arată că ecuaţia (1.67) este validă pentru $n = 1, 2, \ldots$ etc. Astfel teorema este demonstrată.

□

Din (1.67) rezultă că

$$R_{n+1}^{-1} = U_{n+1} D_{n+1}^{-1} U_{n+1}^{\mathrm{T}} \tag{1.71}$$

şi deci ALD calculează într-adevăr descompunerea UDU^{T} a matricei inverse de covarianţă.

Pentru a încheia acest subparagraf să mai notăm că $U_{n+1} D_{n+1}^{-1/2}$ este *factorul Cholesky* inferior triunghiular al matricei inverse R_{n+1}^{-1}.

1.4.2. Testarea pozitivităţii matricelor Toeplitz

Utilizarea sugerată în titlul acestui subparagraf este, mai exact formulată, următoarea: Fie R_{n+1} o matrice Toeplitz dată; determinaţi dacă R_{n+1} este pozitiv definită sau nu. Acest tip

Scanned with OKEN Scanner

de problemă apare într-un număr relativ mare de aplicaţii din domeniile identificării sistemelor şi prelucrării semnalelor[23],[48],[76]. ALD este o procedură numerică eficientă care poate fi utilizată pentru soluţionarea acestui tip de problemă. Intr-adevăr, urmează din Teorema 1.3.1. că matricea R_{n+1} este pozitiv definită dacă şi numai dacă,

$$\rho_0 > 0 \quad \text{şi} \quad |k_p| < 1 \quad (p = 1, 2, \ldots, n). \tag{1.72}$$

Procedura de testare a condiţiei $k_{n+1} > 0$, bazată pe calculul parametrilor $\{k_p\}_{p=1}^n$ cu ALD (în $\sim n^2$ operaţii aritmetice) şi verificarea inegalităţilor (1.72), este mult mai eficientă din punct de vedere numeric decât procedura Sylvester de testare a minorilor principali, sau procedura bazată pe calculul valorilor proprii ale lui R_{n+1}.

Este interesant de remarcat în acest context că ALD poate fi interpretat ca o transformare între domeniul relativ complicat al secvenţelor $\{\rho_k\}_{k=0}^n$ pozitiv definite (domeniu care este un con convex) şi domeniul foarte simplu (un hipercub!) al parametrilor $\{k_p\}_{p=1}^n$.

1.4.3. Testarea stabilităţii

Următoarea problemă este omniprezentă în aplicaţiile teoriei sistemelor, filtrelor numerice şi prelucrării semnalelor: Fie $A_n(z)$ un polinom dat, de grad n; testaţi dacă $A_n(z)$ este stabil, adică dacă

$$z^n A_n(z^{-1}) = 0 \Rightarrow |z| < 1. \tag{1.73}$$

Teorema 1.3.1. sugerează că ALD ar putea fi utilizat pentru testarea stabilităţii unui polinom dat $A_n(z)$. Totuşi, pentru a putea utiliza ALD în acest scop ar trebui să-i reorganizăm calculele într-un mod care este explicat în cele ce urmează.

ALD este o transformare de la secvenţa de covarianţe $\{\rho_k\}_{k=1}^n$ la coeficienţii modelului AR $\{\theta_n\}$. Pentru a testa stabilitatea unui polinom dat $A_n(z)$, utilizând echivalenţa (i) $\Leftrightarrow$ (iii) a Teoremei 1.3.1., ne-ar trebui *transformarea inversă* de la $\{\theta_n\}$ la $\{k_p\}_{p=1}^n$. Această transformare poate fi determinată relativ uşor, după cum se arată mai jos.

Să introducem următoarea notaţie pentru "polinomul reciproc"

$$A^*(z) = z^n A_n(z^{-1}), \quad n = 1, 2, \ldots \tag{1.74}$$

Cu această notaţie, ecuaţia (1.41) se scrie:

$$A_{p+1}(z) = A_p(z) + k_{p+1} z A_p^*(z). \tag{1.75}$$

Din (1.75), prin schimbarea lui z cu z^{-1} şi înmulţirea ambilor membri cu z^{p+1}, se obţine:

$$A_{p+1}^*(z) = k_{p+1} A_p(z) + z A_p^*(z). \tag{1.76}$$

Cele două ecuaţii (1.75) şi (1.76) pot fi scrise sub următoarea formă compactă,

$$\begin{bmatrix} A_{p+1}(z) \\ A_{p+1}^*(z) \end{bmatrix} = \begin{bmatrix} 1 & z k_{p+1} \\ k_{p+1} & z \end{bmatrix} \begin{bmatrix} A_p(z) \\ A_p^*(z) \end{bmatrix}. \tag{1.77}$$

Presupunând $k_{p+1} \neq 1$ (condiţie necesară şi suficientă pentru funcţionarea "normală" a ALD), transformarea (1.77) poate fi inversată pentru a obţine:

$$\begin{bmatrix} A_p(z) \\ A_p^*(z) \end{bmatrix} = \frac{1}{z(1 - k_{p+1}^2)} \begin{bmatrix} z & -z k_{p+1} \\ -k_{p+1} & 1 \end{bmatrix} \begin{bmatrix} A_{p+1}(z) \\ A_{p+1}^*(z) \end{bmatrix}, \tag{1.78}$$

Scanned with OKEN Scanner

sau, echivalent,

$$A_p(z) = \frac{A_{p+1}(z) - k_{p+1}A^*_{p+1}(z)}{1 - k^2_{p+1}}. \tag{1.79}$$

Ecuaţia polinomială (1.79) poate fi iterată pentru a determina secvenţa $\{k_p\}_{p=1}^n$ corespunzătoare unui polinom dat $A_n(z)$.(Observaţie: k_n = coeficientul lui z^n din $A_n(z)$). În conformitate cu Teorema 1.3.1., polinomul $A_n(z)$ satisface (1.73) (adică este stabil) dacă şi numai dacă secvenţa asociată $\{k_p\}$ determinată cu (1.79) satisface

$$|k_p| < 1, \quad p = 1, \ldots, n. \tag{1.80}$$

Este interesant de remarcat că testul de stabilitate bazat pe ALD, descris mai sus, este o posibilă implementare a procedurii Schur–Cohn concepută special pentru testarea stabilităţii sistemelor discrete [3],[12],[52],[60],[81].

Scanned with OKEN Scanner

Capitolul 2

Algoritmii Delsarte–Genin şi Schur pentru Modele AR Monovariabile

2.1. Algoritmul Delsarte–Genin

Primul paragraf al acestui capitol descrie un algoritm rapid pentru rezolvarea recursivă–în–ordin a sistemului de ecuaţii liniare (1.19), propus recent în [9],[10]. Acest algoritm este mai eficient din punct de vedere numeric decât ALD. El va fi numit algoritmul Delsarte–Genin (ADG), după numele celor care l-au introdus în literatura de specialitate [9]. În afara lui [9] există şi alte referinţe care au ca subiect principal ADG şi pe care cititorul le poate consulta pentru a-şi completa imaginea asupra acestui algoritm şi a utilizărilor sale posibile: [29],[30]. În continuare vom prezenta o deducere simplificată a ADG [29]; deducerea originală a algoritmului [9] este mult mai complexă şi mai dificil de urmărit. Fie Δ_n un vector $(n+1)$–dimensional definit prin următoarea ecuaţie:

$$R_{n+1}\Delta_n = \beta_n \begin{bmatrix} 1 \\ . \\ . \\ . \\ 1 \end{bmatrix}, \tag{2.1}$$

unde β_n este un scalar deocamdată neprecizat.

În mod clar, efectul lui β_n asupra componentelor lui Δ_n este de simplă scalare. Să mai observăm că Δ_n este definit în mod unic prin (2.1) (deoarece R_{n+1} este presupus nesingular) şi că Δ_n este un vector simetric, adică

$$\Delta_n = \tilde{\Delta}_n. \tag{2.2}$$

Proprietatea de simetrie (2.2) a lui Δ_n rezultă imediat din Lema 1.2.1. şi relaţia de definiţie (2.1); această proprietate se va dovedi a fi cheia reducerii numărului de operaţii aritmetice necesitate de ALD pentru rezolvarea sistemului de ecuaţii (1.19). Fie, de asemenea,

$$\gamma_n = r_{n+1}^{\mathrm{T}}\Delta_n = \tilde{r}_{n+1}^{\mathrm{T}}\Delta_n. \tag{2.3}$$

Scanned with OKEN Scanner

Utilizând (2.1) şi (2.3) putem scrie:

$$R_{n+1}\begin{bmatrix}0\\ \Delta_{n-1}\end{bmatrix}=\begin{bmatrix}\rho_0 & r_n^{\mathrm{T}}\\ r_n & R_n\end{bmatrix}\begin{bmatrix}0\\ \Delta_{n-1}\end{bmatrix}=\begin{bmatrix}\gamma_{n-1}\\ \beta_{n-1}\\ \cdot\\ \cdot\\ \cdot\\ \beta_{n-1}\end{bmatrix}. \tag{2.4}$$

Sistemele de ecuaţii (2.1) şi (2.4) pot fi combinate pentru a obţine un sistem cu structura lui (1.19). Să definim

$$\lambda_n=\frac{\beta_n}{\beta_{n-1}}. \tag{2.5}$$

Utilizând (2.1),(2.4) şi (2.5), obţinem:

$$R_{n+1}\left\{\Delta_n-\lambda_n\begin{bmatrix}0\\ \Delta_{n-1}\end{bmatrix}\right\}=\begin{bmatrix}\beta_n-\lambda_n\gamma_{n-1}\\ 0\\ \cdot\\ \cdot\\ \cdot\\ 0\end{bmatrix}. \tag{2.6}$$

Aşa cum se va arăta mai jos, scalarul β_n poate fi totdeauna ales astfel încât primul element al lui Δ_n să fie egat cu unu:

$$(\Delta_n)_1=1. \tag{2.7}$$

Rezultă atunci că sistemul de ecuaţii (2.6) are exact structura lui (1.19). Cum soluţia lui (1.19) este unică, din compararea celor două sisteme obţinem:

$$\begin{bmatrix}1\\ \theta_n\end{bmatrix}=\Delta_n-\lambda_n\begin{bmatrix}0\\ \Delta_{n-1}\end{bmatrix}, \tag{2.8}$$

$$\sigma_n=\beta_n-\lambda_n\gamma_{n-1}. \tag{2.9}$$

Să mai observăm că ultimul element al lui $\{\Delta_n\}$ este tot unu (datorită simetriei acestor vectori). Această observaţie ne permite să deducem o expresie şi pentru k_n = ultimul element al lui θ_n (vezi (1.26)). Intr-adevăr, urmează imediat din (2.8) că

$$k_n=1-\lambda_n. \tag{2.10}$$

Relaţiile (2.8)–(2.10) exprimă parametrii de bază ai ALD $\{\theta_n,\sigma_n,k_n\}$ ca funcţii de $\{\Delta_n,\beta_n\}$. Problema care rămâne este deducerea unor relaţii recursive simple pentru calcularea $\{\Delta_n,\beta_n\}$ şi, de asemenea, demonstrarea proprietăţii (2.7). Teorema următoare are ca subiect exact această problemă.

Teorema 2.1.1. *Fie*

$$\alpha_n=\frac{\beta_{n-1}-\gamma_{n-1}}{\beta_{n-2}-\gamma_{n-2}} \tag{2.11}$$

şi fie $\{\beta_n\}$ *determinat de următoarea ecuaţie recursivă*

$$\beta_n=2\beta_{n-1}-\alpha_n\beta_{n-2} \tag{2.12}$$

Scanned with OKEN Scanner

cu valori inițiale $\beta_0 = \rho_0$ *și* $\beta_1 = \rho_0 + \rho_1$.

Atunci secvența de vectori $\{\Delta_n\}$ *definiți prin (2.1), satisface (2.7) și următoarea ecuație recursivă:*

$$\Delta_n = \begin{bmatrix} \Delta_{n-1} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta_{n-1} \end{bmatrix} - \alpha_n \begin{bmatrix} 0 \\ \Delta_{n-2} \\ 0 \end{bmatrix}. \tag{2.13}$$

Demonstrație: Lema 1.2.1. aplicată lui (2.4) produce următoarea ecuație:

$$R_{n+1} \begin{bmatrix} \Delta_{n-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \beta_{n-1} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{n-1} \\ \gamma_{n-1} \end{bmatrix}. \tag{2.14}$$

În plus, să observăm că

$$R_{n+1} \begin{bmatrix} 0 \\ \Delta_{n-2} \\ 0 \end{bmatrix} = \begin{bmatrix} \rho_0 & r_{n-1}^{\mathrm{T}} & \rho_n \\ r_{n-1} & R_{n-1} & \tilde{r}_{n-1} \\ \rho_n & \tilde{r}_{n-1}^{\mathrm{T}} & \rho_0 \end{bmatrix} \begin{bmatrix} 0 \\ \Delta_{n-2} \\ 0 \end{bmatrix} = \begin{bmatrix} \gamma_{n-2} \\ \beta_{n-2} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{n-2} \\ \gamma_{n-2} \end{bmatrix}. \tag{2.15}$$

Să combinăm membrii drepți ai ecuațiilor (2.4),(2.14) și (2.15) în următorul mod:

$$\begin{bmatrix} \gamma_{n-1} \\ \beta_{n-1} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{n-1} \end{bmatrix} + \begin{bmatrix} \beta_{n-1} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{n-1} \\ \gamma_{n-1} \end{bmatrix} - \alpha_n \begin{bmatrix} \gamma_{n-2} \\ \beta_{n-2} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{n-2} \\ \gamma_{n-2} \end{bmatrix} = \begin{bmatrix} \gamma_{n-1} + \beta_{n-1} - \alpha_n \gamma_{n-2} \\ 2\beta_{n-1} - \alpha_n \beta_{n-2} \\ \cdot \\ \cdot \\ \cdot \\ 2\beta_{n-1} - \alpha_n \beta_{n-2} \\ \gamma_{n-1} + \beta_{n-1} - \alpha_n \gamma_{n-2} \end{bmatrix} = \beta_n \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}. \tag{2.16}$$

Pentru a obține ultima egalitate din (2.16) am utilizat relația (2.11) de definiție a lui α_n și ecuația recursivă (2.12) satisfăcută de secvența $\{\beta_n\}$.

Din (2.4), (2.14)–(2.16) rezultă că

$$R_{n+1} \left\{ \begin{bmatrix} \Delta_{n-1} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta_{n-1} \end{bmatrix} - \alpha_n \begin{bmatrix} 0 \\ \Delta_{n-2} \\ 0 \end{bmatrix} \right\} = \beta_n \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}. \tag{2.17}$$

Comparând (2.17) cu (2.1) și ținând cont de unicitatea soluției Δ_n a sistemului de ecuații (2.1), obținem imediat relația recursivă (2.13) pentru Δ_n.

Scanned with OKEN Scanner

Ecuaţie	Nr. operaţii (·) sau (:)	Nr. operaţii (+)
Iniţializare		
$\Delta_0 = 1, \beta_0 = \rho_0, \gamma_0 = \rho_1$	–	–
$\Delta_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \beta_1 = \rho_0 + \rho_1, \quad \gamma_1 = \rho_1 + \rho_2$	–	$\mathcal{O}(1)$
Pentru $n = 2, \ldots, n_{max}$ calculează:		
A.		
$\alpha_n = \frac{\beta_{n-1} - \gamma_{n-1}}{\beta_{n-2} - \gamma_{n-2}}$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\beta_n = 2\beta_{n-1} - \alpha_n \beta_{n-2}$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\Delta_n = \begin{bmatrix} \Delta_{n-1} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta_{n-1} \end{bmatrix} - \alpha_n \begin{bmatrix} 0 \\ \Delta_{n-2} \\ 0 \end{bmatrix}$	$\sim \frac{n}{2}$	$\sim n$
$\gamma_n = r_{n+1}^{\mathrm{T}} \Delta_n = (\rho_1 + \rho_{n+1}) + \Delta_{n,1}(\rho_2 + \rho_n) + \ldots$	$\sim \frac{n}{2}$	$\sim n$
B.		
$\lambda_n = \frac{\beta_n}{\beta_{n-1}}$	$\mathcal{O}(1)$	–
$\sigma_n = \beta_n - \lambda_n \gamma_{n-1}$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$k_n = 1 - \lambda_n$	–	$\mathcal{O}(1)$
C.		
$\begin{bmatrix} 1 \\ \theta_n \end{bmatrix} = \Delta_n - \lambda_n \begin{bmatrix} 0 \\ \Delta_{n-1} \end{bmatrix}$	$\sim \frac{n}{2}$	$\sim n$

Tabelul 2.1. Algoritmul Delsarte-Genin.

Mai rămâne să arătăm că Δ_n satisface (2.7). Din ecuaţia (2.13) observăm că primul (ultimul) element al lui Δ_n este acelaşi cu primul (ultimul) element al lui Δ_{n-1}. Este astfel suficient să arătăm că $\Delta_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Totuşi, calcule simple arată că pentru $\beta_0 = \rho_0$ şi $\beta_1 = \rho_0 + \rho_1$ (utilizate ca valori iniţiale pentru ecuaţia recurentă (2.12)), corespund următoarele valori ale lui Δ_0 şi Δ_1:

$$\begin{aligned} \Delta_0 &= 1, \\ \Delta_1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \end{aligned}$$

Cu această observaţie, demonstraţia este terminată.

□

ADG este constituit din ecuaţiile (2.5),(2.8)–(2.13). Aceste ecuaţii includ relaţii recurente cu 3 termeni şi par mai complicate decât ecuaţiile corespunzătoare ALD. În realitate ele pot fi implementate mai eficient decât ecuaţiile ALD, *datorită simetriei vectorilor* $\{\Delta_n\}$.

Ecuaţiile ADG sunt prezentate sintetic în Tabelul 2.1., împreună cu numărul aproximativ de operaţii aritmetice necesare pentru implementare. Există două moduri în care ADG poate fi implementat, în funcţie de aplicaţia avută în vedere.

Scanned with OKEN Scanner

a) Dacă nu se doresc toţi vectorii $\{\theta_n\}_{n=1}^{n_{max}}$, ci numai $\theta_{n_{max}}$, atunci numai grupurile de ecuaţii (A) şi (B) din Tabelul 2.1. trebuie iterate pentru $n = 2, \ldots, n_{max}$. Ecuaţia matriceală a grupului (C) va fi utilizată numai pentru $n = n_{max}$. Numărul aproximativ de operaţii cerute de acest mod de implementare a ADG este următorul:

$$N_r(\cdot) \approx \sum_{n=1}^{n_{max}} n \approx \frac{n_{max}^2}{2} \approx \frac{1}{2} N_r(\cdot) \text{ necesare ALD,}$$

$$N_r(+) \approx 2 \sum_{n=1}^{n_{max}} n \approx n_{max}^2 \approx N_r(+) \text{ necesare ALD.}$$

Deoarece pe multe tipuri de calculatoare numerice, o înmulţire se efectuează mult mai lent decât o adunare, rezultă că *ADG implementat ca mai sus este de aproximativ două ori mai rapid decât ALD.*

Să remarcăm că evaluarea grupului (B) de ecuaţii nu este necesară pentru iterarea ecuaţiilor din grupul (A). Astfel, grupul (B) de ecuaţii ar putea, la fel ca (C), să fie utilizat numai pentru $n = n_{max}$. Totuşi ecuaţiile grupului (B) cer un număr insignificant de operaţii pentru implementare. În plus, secvenţa $\{\sigma_n\}_{n=1}^{n_{max}}$ este necesară în modelarea AR pentru determinarea ordinului modelului[75],[76], iar secvenţa $\{k_n\}_{n=1}^{n_{max}}$ poate, de asemenea, să fie utilă (de exemplu, pentru a verifica stabilitatea modelului AR determinat, vezi Capitolul 1). Datorită motivelor anterioare,evaluarea ecuaţiilor grupului (B) este deseori recomandabil să se facă pentru $n = 1, \ldots, n_{max}$, împreună cu ecuaţiile grupului (A).

b) Dacă se doresc toţi vectorii $\{\theta_n\}_{n=1}^{n_{max}}$ (de exemplu, într-o aplicaţie în care este necesară determinarea factorului Cholesky al matricei inverse de covarianţă, vezi Capitolul 1), atunci numărul aproximativ de operaţii necesare implementării ADG este următorul:

$$N_r(\cdot) \approx \frac{3}{2} \sum_{n=1}^{n_{max}} n \approx 0,75 n_{max}^2 \approx 0,75 N_r(\cdot) \text{ necesare ALD,}$$

$$N_r(+) \approx 3 \sum_{n=1}^{n_{max}} n \approx 1,5 n_{max}^2 \approx 1,5 N_r(+) \text{ necesare ALD.}$$

Se observă că ADG îşi pierde din eficienţă în acest caz, după cum era de aşteptat. Totuşi, pe calculatoarele pe care o adunare se efectuează mult mai eficient decât o înmulţire, ADG rămâne de aproximativ $\frac{4}{3}$ ori mai rapid decât ALD.

Inainte de a încheia acest paragraf, să notăm că toate comparaţiile anterioare între ALD şi ADG sunt valide pentru n suficient de mare (de exemplu, $n \geq 10$). Pentru valori mici ale lui n, termenii $\mathcal{O}(n)$ din $N_r(\cdot)$ şi $N_r(+)$ nu mai pot fi neglijaţi şi, ca o consecinţă, se poate arăta că ADG devine mai puţin eficient decât ALD. Să nu uităm, totuşi, că algoritmii rapizi sunt frecvent utilizaţi în aplicaţiile cu $n \gg 1$ şi nu în cele cu n de ordinul unităţilor (să zicem $n < 10$). În astfel de aplicaţii cu $n \gg 1$ îşi demonstrează ADG întregul său potenţial şi, în particular, eficienţa sa superioară ALD.

2.2. Algoritmul Schur

Algoritmul Levinson–Durbin poate fi utilizat, printre altele, pentru a calcula factorizarea Cholesky a matricei *inverse* de covarianţă, vezi Capitolul 1. În anumite aplicaţii apare necesară determinarea factorului Cholesky al matricei de covarianţă însăşi. Dacă acest factor este calculat

Scanned with OKEN Scanner

utilizând algoritmi standard care nu exploatează structura specială de tip Toeplitz a matricei de covarianţă, atunci numărul de operaţii aritmetice necesare este $\mathcal{O}(m^3)$, unde m este dimensiunea matricei considerate. Dacă structura Toeplitz a matricei de covarianţă este exploatată, atunci factorul Cholesky al acesteia poate fi calculat în un număr de operaţii cu un ordin de mărime mai mic: $\mathcal{O}(m^2)$.

Algoritmul Schur este un algoritm rapid care efecuează factorizarea Cholesky a unei matrice de covarianţă date, R_{m+1}, în $\mathcal{O}(m^2)$ operaţii aritmetice[13],[36],[42],[62]. Prezentul paragraf deduce algoritmul Schur (AS) şi discută utilizarea lui atât ca o procedură utilizabilă pentru a testa dacă o matrice Toeplitz dată, R_{m+1}, este pozitiv definită sau nu, cât şi pentru determinarea factorizării Cholesky a lui R_{m+1}, când aceasta există (sau, echivalent, când $R_{m+1} > 0$).

Factorizarea inferior–superior triunghiulară (1.71) a matricei inverse de covarianţă poate fi utilizată ca un punct de pornire în deducerea AS. Totuşi, vom prefera să utilizăm factorizarea superior–inferior triunghiulară, din două motive: (a) pentru a introduce şi acest tip de factorizare şi a releva modul simplu în care ea se deduce din (1.71); (b) pentru a simplifica, într-o oarecare măsură , deducerea AS.

Următoarea lemă deduce factorizarea superior–triunghiulară a lui R_{m+1}^{-1} din factorizarea inferior–superior triunghiulară.

Lema 2.2.1. *Cantităţile* $\{\theta_n\}, R_{m+1}$ *şi* $\{\sigma_n\}$, *introduse în Capitolul 1, satisfac următoarea ecuaţie de factorizare:*

$$V_{m+1}^{\mathrm{T}} R_{m+1} V_{m+1} = S_{m+1}, \tag{2.18}$$

unde

$$S_{m+1} = \begin{bmatrix} \sigma_0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \sigma_1 & 0 & \cdot & & 0 \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & & & \cdot & \cdot & \\ \cdot & & & & \cdot & 0 \\ 0 & \cdot & \cdot & & 0 & \sigma_m \end{bmatrix} \quad (\sigma_0 \triangleq \rho_0), \tag{2.19}$$

$$V_{m+1} = \begin{bmatrix} 1 & \tilde{\theta}_1 & & & & \\ 0 & 1 & & \cdot & & \\ \cdot & \cdot & & & \cdot & \\ \cdot & & \cdot & & & \tilde{\theta}_m \\ \cdot & & & \cdot & 1 & \\ 0 & \cdot & \cdot & & 0 & 1 \end{bmatrix}. \tag{2.20}$$

Demonstraţie: Pentru a demonstra această lemă, să introducem "matricea de permutare"

$$J = \begin{bmatrix} 0 & \cdot & \cdot & \cdot & 0 & 1 \\ 0 & & & & 1 & 0 \\ \cdot & & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ 0 & 1 & 0 & & & \cdot \\ 1 & 0 & \cdot & \cdot & \cdot & 0 \end{bmatrix} \tag{2.21}$$

de dimensiuni $(m+1) \times (m+1)$ şi să observăm că

$$JJ = I$$

Scanned with OKEN Scanner

$$\begin{aligned} JR_{m+1}J &= R_{m+1} \\ JD_{m+1}J &= S_{m+1} \\ JU_{m+1}J &= V_{m+1}. \end{aligned}$$

Utilizând aceste observaţii şi (1.67), putem scrie:

$$\begin{aligned} V_{m+1}^{\mathrm{T}} R_{m+1} V_{m+1} &= (JU_{m+1}J)^{\mathrm{T}}(JR_{m+1}J)(JU_{m+1}J) = JU_{m+1}^{\mathrm{T}} R_{m+1} U_{m+1} J = JD_{m+1}J = \\ &= S_{m+1} \end{aligned} \tag{2.22}$$

şi demonstraţia este terminată.

□

Din (2.18) obţinem următoarea factorizare a lui R_{m+1}:

$$R_{m+1} = H_{m+1}^{\mathrm{T}} S_{m+1} H_{m+1}, \tag{2.23}$$

unde

$$H_{m+1}^{\mathrm{T}} = V_{m+1}^{-\mathrm{T}}. \tag{2.24}$$

Problema principală considerată în continuare este calculul rapid (în $\mathcal{O}(m^2)$ operaţii) al factorului Cholesky $S_{m+1}^{1/2} H_{m+1}$ al lui R_{m+1}. Pentru a rezolva această problemă, să observăm mai întâi că H_{m+1}^{T} este o matrice inferior triunghiulară cu unu pe diagonală

$$H_{m+1}^{\mathrm{T}} \triangleq \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 \\ & 1 & \cdot & & & \cdot \\ h_m & & \cdot & \cdot & & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & & 1 & 0 \\ & & & & h_1 & 1 \end{bmatrix}, \tag{2.25}$$

unde h_k este un vector k dimensional. Să observăm, de asemenea, următoarele relaţii importante între coloanele corespunzătoare ale lui H_{m+1}^{T} şi V_{m+1}:

$$\begin{aligned} R_{m+1} \begin{bmatrix} \tilde{\theta}_n \\ 1 \\ 0 \end{bmatrix} &= V_{m+1}^{-\mathrm{T}} S_{m+1} V_{m+1}^{-1} \begin{bmatrix} \tilde{\theta}_n \\ 1 \\ 0 \end{bmatrix} = \\ &= H_{m+1}^{\mathrm{T}} S_{m+1} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \sigma_n \begin{bmatrix} 0 \\ 1 \\ h_{m-n} \end{bmatrix}, \quad n = 0, \ldots, m. \end{aligned} \tag{2.26}$$

Astfel,

$$\begin{bmatrix} \rho_0 \\ \cdot \\ \cdot \\ \cdot \\ \rho_m \end{bmatrix} = \sigma_0 \begin{bmatrix} 1 \\ h_m \end{bmatrix} \tag{2.27}$$

şi

$$R_{m+1} \begin{bmatrix} \tilde{\Delta}_{n+1} \\ 0 \end{bmatrix} = \sigma_{n+1} \begin{bmatrix} 0 \\ 1 \\ h_{m-n-1} \end{bmatrix}, \quad n = 0, \ldots, m+1, \tag{2.28}$$

Scanned with OKEN Scanner

unde

$$\Delta_{n+1} = \begin{bmatrix} 1 \\ \theta_{n+1} \end{bmatrix} \tag{2.29}$$

Ideea de bază în deducerea AS poate fi rezumată astfel: utilizând ecuaţiile algoritmului Levinson–Durbin, Δ_{n+1} poate fi exprimat în funcţie de Δ_n; atunci, utilizând (2.28), putem spera să-l exprimăm pe h_{m-n-1} în funcţie de h_{m-n} obţinând astfel un algoritm recursiv de calcul al $\{h_k\}$, pentru a cărui iniţializare putem utiliza (2.27). Fezabilitatea ideii anterioare va fi demonstrată în cele ce urmează. Totuşi, să notăm că implementarea ei nu este aşa de directă cum ar putea să pară, datorită necesităţii introducerii unei secvenţe de vectori *auxiliari* $\{g_k\}$ de aceeaşi dimensiune ca h_k. Aceşti vectori $\{g_k\}$ apar în mod firesc atunci când se încearcă exprimarea lui $\{h_k\}$ în funcţie de $\{h_{k+1}\}$ (vezi mai jos). Elementul g_{m-n-1} al secvenţei de vectori auxiliari $\{g_k\}$ este definit prin următoarea ecuaţie:

$$R_{m+1} \begin{bmatrix} \Delta_{n+1} \\ 0 \end{bmatrix} = \sigma_{n+1} \begin{bmatrix} 1 \\ 0 \\ g_{m-n-1} \end{bmatrix}. \tag{2.30}$$

Faptul că primele $n+2$ elemente ale membrului drept al ecuaţiei (2.30) au expresia

$$\sigma_{n+1} \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T,$$

este o consecinţă directă a ecuaţiei (1.19).

După aceste preparative, suntem pregătiţi să demonstrăm rezultatul principal al acestui paragraf.

Teorema 2.2.1. *Elementele* $\{h_k\}$ *şi* $\{\sigma_k\}$ *ale factorizării Cholesky* (2.23) *a lui* R_{m+1} *satisfac următoarele ecuaţii recursive* $(n = 0, 1, \dots, m-1)$*:*

$$\sigma_{n+1} h_{m-n-1} = \sigma_n \underline{h}_{m-n} + k_{n+1} \sigma_n \overline{g}_{m-n} \tag{2.31}$$

$$\sigma_{n+1} g_{m-n-1} = \sigma_n \overline{g}_{m-n} + k_{n+1} \sigma_n \underline{h}_{m-n} \tag{2.32}$$

$$k_{n+1} = -(g_{m-n})_1 \tag{2.33}$$

$$\sigma_{n+1} = \sigma_n (1 - k_{n+1}^2) \tag{2.34}$$

unde $\underline{h}_{m-n}$ *şi* $\overline{g}_{m-n}$ *sunt vectori* $m-n-1$ *dimensionali obţinuţi din* h_{m-n} *şi* g_{m-n} *prin eliminarea ultimului, respectiv, primului element; iar* $(g_{m-n})_1$ *este primul element al vectorului* g_{m-n}.

Demonstraţie: Să rescriem ecuaţiile (2.28) şi (2.30) într-o formă mai detaliată:

$$\left[\begin{array}{cccc} \rho_0 & \rho_1 & \dots & \rho_{n+1} \\ \rho_1 & \rho_0 & & \rho_n \\ \vdots & & & \vdots \\ \rho_{n+1} & & & \rho_0 \\ \hline \rho_{n+2} & \dots & & \rho_1 \\ \vdots & & & \vdots \\ \rho_m & \dots & & \rho_{m-n-1} \end{array}\right] \tilde{\Delta}_{n+1} = \sigma_{n+1} \left[\begin{array}{c} 0 \\ 1 \\ \hline h_{m-n-1} \end{array}\right] \tag{2.35}$$

Scanned with OKEN Scanner

şi, respectiv,

$$\left[\begin{array}{cccc} \rho_0 & \rho_1 & \cdots & \rho_{n+1} \\ \rho_1 & \rho_0 & & \rho_n \\ \vdots & & \ddots & \vdots \\ \rho_{n+1} & & & \rho_0 \\ \hline \rho_{n+2} & \cdots & & \rho_1 \\ \vdots & & & \vdots \\ \rho_m & \cdots & & \rho_{m-n-1} \end{array}\right] \Delta_{n+1} = \sigma_{n+1} \left[\begin{array}{c} 1 \\ 0 \\ \hline g_{m-n-1} \end{array}\right]. \tag{2.36}$$

Ecuaţia de bază (1.26) a ALD implică următoarea relaţie recurentă pentru secvenţa de vectori $\{\Delta_n\}$:

$$\Delta_{n+1} = \begin{bmatrix} \Delta_n \\ 0 \end{bmatrix} + k_{n+1} \begin{bmatrix} 0 \\ \tilde{\Delta}_n \end{bmatrix}. \tag{2.37}$$

Prin aplicarea "operatorului de reversare" " ~ " ecuaţiei (2.37) obţinem:

$$\tilde{\Delta}_{n+1} = \begin{bmatrix} 0 \\ \tilde{\Delta}_n \end{bmatrix} + k_{n+1} \begin{bmatrix} \Delta_n \\ 0 \end{bmatrix}. \tag{2.38}$$

Din combinarea ecuaţiilor (2.35) cu (2.38) şi (2.36) cu (2.37) rezultă, după unele calcule simple, următoarele relaţii esenţiale:

$$\sigma_{n+1} \left[\begin{array}{c} 0 \\ 1 \\ \hline h_{m-n-1} \end{array}\right] = \sigma_n \left[\begin{array}{c} r_{n+1}^{\mathrm{T}} \tilde{\Delta}_n / \sigma_n \\ 0 \\ \hline \underline{h}_{m-n} \end{array}\right] + \sigma_n k_{n+1} \left[\begin{array}{c} 1 \\ 0 \\ \hline (g_{m-n})_1 \\ g_{m-n} \end{array}\right] \tag{2.39}$$

şi, respectiv,

$$\sigma_{n+1} \left[\begin{array}{c} 1 \\ 0 \\ \hline h_{m-n-1} \end{array}\right] = \sigma_n \left[\begin{array}{c} 1 \\ 0 \\ \hline (g_{m-n})_1 \\ g_{m-n} \end{array}\right] + \sigma_n k_{n+1} \left[\begin{array}{c} r_{n+1}^{\mathrm{T}} \tilde{\Delta}_n / \sigma_n \\ 0 \\ \hline \underline{h}_{m-n} \end{array}\right]. \tag{2.40}$$

Se observă acum foarte clar că ecuaţiile pentru calculul recurent al $\{h_k\}$ şi $\{g_k\}$ sunt *cuplate* şi, deci, introducerea vectorilor auxiliari $\{g_k\}$ este într-adevăr necesară calculului recursiv al vectorilor $\{h_k\}$ ai factorizării Cholesky.

Din (2.39) şi (2.40) se obţin imediat următoarele ecuaţii: (2.31),(2.32),(2.33),(2.34) şi, în plus,

$$k_{n+1} = -\frac{r_{n+1}^{\mathrm{T}} \tilde{\Delta}_n}{\sigma_n} = -\frac{\rho_{n+1} + r_n^{\mathrm{T}} \tilde{\theta}_n}{\sigma_n} \tag{2.41}$$

care coincide cu ecuaţiile (1.21),(1.22) din Capitiolul 1; să notăm că ecuaţia (2.41) nu va fi utilizată în prezentul context (evaluarea lui k_{n+1} cu (2.41) ar cere cunoaşterea lui θ_n – cantitate care *nu* se calculează în cadrul AS); totuşi deducerea lui (2.41) şi coincidenţa cu (1.21),(1.22) furnizează o verificare utilă a analizei efectuate. Cu aceste observaţii, demonstraţia este terminată.

Scanned with OKEN Scanner

Ecuaţiile AS deduse în Teorema 2.2.1. sunt prezentate sintetic în Tabelul 2.2. împreună cu numărul aproximativ de operaţii aritmetice necesare pentru implementare. În Tabelul 2.2.:

$$\mu_{m-n} \triangleq \sigma_n h_{m-n}, \tag{2.42}$$

$$\nu_{m-n} \triangleq \sigma_n g_{m-n}, \tag{2.43}$$

iar notaţiile $\underline{\mu}_{m-n}$ şi $\overline{\nu}_{m-n}$ îşi păstrează semnificaţia lor anterioară. Să notăm că propagarea vectorilor $\{\mu_k\}$ şi $\{\nu_k\}$ în locul lui $\{h_k\}$ şi $\{g_k\}$ se poate face mult mai economicos: la iteraţia n, se salvează, în acest mod, circa $2(m-n-1)$ înmulţiri; deci, în total, se elimină aproximativ m^2 înmulţiri. Desigur, procedând astfel, nu vom calcula coloanele lui H_{m+1}^{T} ci ale lui $H_{m+1}^{\mathrm{T}} S_{m+1}$.

Pentru multe aplicaţii calculul factorului Cholesky "normalizat" $H_{m+1}^{\mathrm{T}} S_{m+1}$ este suficient. Totuşi, dacă se doreşte evaluarea lui H_{m+1} în mod explicit, aceasta se poate face uşor în circa $m^2/2$ înmulţiri (utilizând pentru aceasta matricele $H_{m+1}^{\mathrm{T}} S_{m+1}$ şi S_{m+1}, ambele furnizate de AS din Tabelul 2.2.). Observaţi că procedând ca mai sus pentru evaluarea explicită a lui H_{m+1}, se salvează $m^2/2$ înmulţiri faţă de cazul în care AS propagă direct $\{h_k\}$ şi $\{g_k\}$, motivul fiind, desigur, că se elimină "scalarea" lui g_{m-n-1} (adică înmulţirea acestui vector cu σ_{n+1}/σ_n) la fiecare iteraţie.

Numărul total aproximativ de operaţii aritmetice cerute de implementarea AS din Tabelul 2.2. este următorul

$$2\sum_{n=1}^{m}(m-n) \approx m^2 \quad [flop]. \tag{2.44}$$

Astfel, AS şi ALD au aceeaşi complexitate numerică (comparaţi (2.44) şi (1.33)). Spre deosebire de ALD, algoritmul Schur nu calculează vectorii $\{\theta_n\}$ şi deci nu poate fi utilizat în modelarea AR în aplicaţiile în care se cere parametrizarea modelului prin $\{\theta_n\}$. Totuşi, AS determină secvenţa $\{k_n\}$ care, de asemenea, poate fi utilizată pentru a parametriza un model AR (de fapt în multe aplicaţii din domeniul prelucrării semnalelor, parametrizarea prin $\{k_n\}$ este preferată celei prin $\{\theta_n\}$)[23],[40],[45],[48]. Mai mult, AS determină această secvenţă $\{k_n\}$ cu o *precizie numerică superioară* celei corespunzătoare ALD. Acest fapt stabilit prin studii atât empirice, cât şi analitice, poate fi explicat relativ uşor. Implementările numerice (în special cele în virgulă fixă) ale unui algoritm sunt cu atât mai afectate de erori de rotunjire cu cât variabilele algoritmului au domenii mai diferite de variaţie. Să considerăm mai întâi ALD. Să presupunem că matricea $R_{m+1} > 0$ a fost scalată prin împărţirea cu ρ_0; astfel $(R_{m+1})_{ii} = 1$. Urmează atunci din proprietăţile şi ecuaţiile ALD că $|k_n| < 1$ şi $0 < \sigma_n < 1$. Totuşi elementele lui θ_n pot avea valori mult mai mari decât 1, în modul.

Astfel domeniile de variaţie ale variabilelor ALD pot fi foarte diferite. Variabilele AS, în schimb, variază toate în plaja $(-1,+1)$, după cum se arată în următoarea lemă.

Lema 2.2.2. *Fie γ o variabilă generică a AS. Atunci*

$$|\gamma| < 1. \tag{2.45}$$

Demonstraţie: Pentru $\gamma = k_n$ şi $\gamma = \sigma_n$, inegalitatea (2.45) urmează din discuţia anterioară despre ALD. Fie $\gamma =$ un element al vectorului ν_{m-n}. Să notăm cu $R_{m+1}^{1/2}$ o rădăcină pătrată a matricei R_{m+1} şi cu $\|\cdot\|$ norma vectorială euclidiană. Din ecuaţia (2.30) rezultă că

$$\gamma = u^{\mathrm{T}} R_{m+1}^{1/2} \begin{bmatrix} \Delta_n \\ 0 \end{bmatrix}, \tag{2.46}$$

Scanned with OKEN Scanner

Ecuaţie	Număr operaţii [flop]
Iniţializare:	
$\sigma_0 = \rho_0; \quad k_1 = -\rho_1/\rho_0$	$\mathcal{O}(1)$
$\mu_m = \left[\begin{array}{ccc} \rho_1 & \dots & \rho_m \end{array} \right]^{\mathrm{T}} = \nu_m$	–
Pentru $n = 0, \dots, m-1$ calculează:	
$\mu_{m-n-1} = \underline{\mu}_{m-n} + k_{n+1}\overline{\nu}_{m-n}$	$\sim (m-n)$
$\nu_{m-n} = \overline{\nu}_{m-n} + k_{n+1}\underline{\mu}_{m-n}$	$\sim (m-n)$
$k_{n+1} = -(\nu_{m-n})_1/\sigma_n$	$\mathcal{O}(1)$
$\sigma_{n+1} = \sigma_n(1-k_{n+1}^2)$	$\mathcal{O}(1)$

Tabelul 2.2. Algoritmul Schur.

unde u^{T}este linia lui $R_{m+1}^{1/2}$ corespunzătoare lui γ. Astfel,

$$|\gamma| \leq ||u|| \cdot ||R_{m+1}^{1/2} \left[\begin{array}{c} \Delta_n \\ 0 \end{array} \right] ||. \tag{2.47}$$

Totuşi $||u|| = 1$ deoarece $u^{\mathrm{T}}u = (R_{m+1})_{ii} = 1$ (pentru un i corespunzător), iar

$$||R_{m+1}^{1/2} \left[\begin{array}{c} \Delta_n \\ 0 \end{array} \right] ||^2 = \left[\begin{array}{cc} \Delta_n^{\mathrm{T}} & 0 \end{array} \right] R_{m+1} \left[\begin{array}{c} \Delta_n \\ 0 \end{array} \right] = \sigma_n \leq 1. \tag{2.48}$$

În concluzie

$$|\gamma| < 1 \tag{2.49}$$

şi demonstraţia este încheiată.

□

Precizia numerică superioară a AS faţă de ALD constituie un avantaj important datorită căruia AS este preferat lui ALD în aplicaţiile în care se doreşte determinarea secvenţei $\{k_n\}$, în special în situaţiile în care matricea R_{m+1} este prost–condiţionată (adică valorile ei proprii sunt puternic dispersate).

În privinţa factorizării calculate, domeniile de aplicaţie ale celor doi algoritmi sunt disjuncte: ALD calculează factorul Cholesky al matricei inverse de covarianţă R_{m+1}^{-1}, în timp ce AS calculează factorul Cholesky al lui R_{m+1}. Astfel, pentru aplicaţiile de factorizare, tipul aplicaţiei va decide care dintre aceşti doi algoritmi trebuie utilizat.

În fine, să remarcăm că eficienţa numerică a AS poate fi crescută utilizând idei similare celor aplicate ALD în primul paragraf al acestui capitol[10],[29].

Scanned with OKEN Scanner

Capitolul 3

Algoritmul Levinson pentru Modele ARMA Monovariabile

3.1. Modele AR–plus–zgomot şi ARMA

Fie $\{y(t) \in \mathcal{R}\}$, $t = 1, 2, \ldots$, un semnal AR generat de următoarea ecuaţie

$$A(q^{-1})y(t) = e(t), \tag{3.1}$$

unde q^{-1} este operatorul de întârziere cu un tact, $A(q^{-1})$ este următorul polinom în q^{-1},

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}, \tag{3.2}$$

iar $e(t)$ este zgomot alb de medie zero şi dispersie σ_e,

$$\begin{aligned} Ee(t) &= 0, \\ Ee(t)e(s) &= \sigma_e \delta_{t,s}. \end{aligned} \tag{3.3}$$

Modelele AR de tipul anterior au fost introduse în Capitolul 1 la care ne vom referi pentru detalii privind utilizările acestor modele, a presupunerilor uzuale făcute asupra lor şi a estimării parametrilor ce descriu acest tip de model. În contrast cu ipotezele făcute în Capitolul 1, vom presupune acum că semnalul AR $y(t)$ nu poate fi măsurat exact; astfel semnalul disponibil este

$$x(t) = y(t) + \varepsilon(t), \tag{3.4}$$

unde $\varepsilon(t)$ este un zgomot alb de măsură, presupus a fi de medie zero, dispersie σ_ε şi necorelat cu $e(t)$:

$$E\varepsilon(t) = 0, \quad E\varepsilon(t)e(s) = 0, \tag{3.5}$$

$$E\varepsilon(t)\varepsilon(s) = \sigma_\varepsilon \delta_{t,s}. \tag{3.6}$$

Combinând (3.1) şi (3.4), obţinem:

$$A(q^{-1})x(t) = e(t) + A(q^{-1})\varepsilon(t). \tag{3.7}$$

Membrul drept al ecuaţiei (3.7) nu mai este zgomot alb (pentru $\sigma_\varepsilon > 0$).Ca o consecinţă a acestui fapt, parametrii $\{a_i\}$ care descriu semnalul AR $y(t)$ nu mai pot fi consistent estimaţi

Scanned with OKEN Scanner

prin metoda CMMP descrisă în Capitolul 1[75],[76]. O metodă similară de estimare poate fi, totuşi, obţinută relativ uşor pornind de la observaţia că pentru $k \geq 1$

$$\begin{aligned} &E[x(t-n-k)][e(t)+A(q^{-1})\varepsilon(t)] = \\ &= E[\varepsilon(t-n-k)+\text{comb. lin. } \{e(t-n-k), e(t-n-k-1), \ldots\}] \cdot \\ &\cdot[e(t)+A(q^{-1})\varepsilon(t)] = 0. \end{aligned} \tag{3.8}$$

Inmulţind ambii membri ai ecuaţiei (3.7) cu $x(t-n-k)$, aplicând operatorul de mediere "E" factorilor rezultaţi şi utilizând (3.8), obţinem imediat următoarele ecuaţii liniare:

$$\rho_{n+k} + a_1\rho_{n+k-1} + \ldots + a_n\rho_k = 0, \quad k \geq 1, \tag{3.9}$$

unde

$$\rho_k = E[x(t)x(t-k)]. \tag{3.10}$$

Să introducem următoarele notaţii similare (dar nu identice!) notaţiilor corespunzătoare folosite în Capitolul 1:

$$R = \begin{bmatrix} \rho_n & \rho_{n-1} & \ldots & \rho_1 \\ \rho_{n+1} & \rho_n & \ldots & \rho_{n-1} \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ \rho_{2n-1} & \rho_{2n-2} & \ldots & \rho_n \end{bmatrix}, \tag{3.11}$$

$$r = \begin{bmatrix} \rho_{n+1} \\ \cdot \\ \cdot \\ \cdot \\ \rho_{2n} \end{bmatrix}. \tag{3.12}$$

Utilizând notaţiile anterioare, putem scrie primele n ecuaţii (3.9) (corespunzătoare lui $k = 1, 2, \ldots, n$) sub următoarea formă compactă:

$$R\theta = -r, \tag{3.13}$$

unde θ este vectorul parametrilor AR de estimat:

$$\theta = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}^T. \tag{3.14}$$

Intrucât matricea R este nesingulară[75],[76], θ este soluţia unică a sistemului de ecuaţii liniare (3.13). În concluzie, putem obţine o estimaţie consistentă a parametrilor $\{a_i\}$ ai semnalului AR neperturbat $y(t)$, rezolvând sistemul liniar (3.13) unde covarianţele teoretice ce apar în R şi r sunt înlocuite prin covarianţe de eşantion (calculate, de exemplu, cu formula (1.15)).

Sistemul de ecuaţii liniare (3.13) poate fi generalizat, într-o anumită direcţie.

Pentru a prezenta această generalizare, să considerăm un semnal $x(t)$ generat de următoarea ecuaţie:

$$A(q^{-1})x(t) = C(q^{-1})e(t), \tag{3.15}$$

Scanned with OKEN Scanner

unde $A(q^{-1})$ şi $e(t)$ sunt definite exact ca mai sus, iar

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_m q^{-m}. \tag{3.16}$$

Ecuaţia (3.15) se numeşte de *autoregresie şi medie alunecătoare* (ARMA).

Când specificarea ordinelor acestei ecuaţii este necesară pentru claritate, ea este desemnată prin abrevierea ARMA(n,m). Justificarea denumirii părţii AR a acestei ecuaţii este similară celei prezentate în paragraful 1.1 pentru modelele AR pure. Partea MA a ecuaţiei ARMA (3.15)

$$C(q^{-1})e(t) = e(t) + c_1 e(t-1) + \ldots + c_m e(t-m) \tag{3.17}$$

poate fi interpretată ca o medie (ponderată) a eşantioanelor $\{e(t), \ldots, e(t-m)\}$; mai mult, această "medie" este "alunecătoare" (glisantă), deoarece variabila timp t variază. Denumirea de MA pentru partea dreaptă a ecuaţiei (3.15) apare, astfel, justificată.

Modelele ARMA au utilizări similare modelelor AR. În unele aplicaţii modelele ARMA sunt preferate celor de tip AR datorită flexibilităţii lor suplimentare (orice ecuaţie AR este un caz special de ecuaţie ARMA, corespunzând lui $C(q^{-1}) \equiv 1$)[76]. Datorită generalităţii lor, modelele ARMA sunt mai dificil de determinat în aplicaţii decât modelele AR. Din fericire, parametrii AR ai unei ecuaţii ARMA pot fi determinaţi relativ uşor (pentru determinarea parametrilor MA $\{c_i\}$, care este o operaţie mai dificilă, ne vom referi la următoarele lucrări [54],[63],[78],[96]). Să observăm că

$$\begin{aligned} Ex(t-m-k)C(q^{-1})e(t) &= E[\text{comb.liniară}\{e(t-m-k), e(t-m-k-1), \ldots\}] \cdot \\ &\cdot [C(q_{-1})e(t)] = 0. \end{aligned} \tag{3.18}$$

Inmulţind ambii membri ai ecuaţiei ARMA (3.15) cu $x(t-m-k)$, aplicând operatorul de mediere şi utilizând (3.18), obţinem ecuaţii cu $\{a_i\}$ ca necunoscute:

$$\rho_{m+k} + a_1 \rho_{m+k-1} + \ldots + a_n \rho_{m+k-n} = 0, \quad k \geq 1. \tag{3.19}$$

Primele n ecuaţii ale setului de ecuaţii liniare (3.19) pot fi scrise sub forma (3.13) unde acum:

$$R = \begin{bmatrix} \rho_m & \rho_{m-1} & \cdots & \rho_{m+1-n} \\ \rho_{m+1} & \rho_m & & \rho_{m-n} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \rho_{m+n-1} & \rho_{m+n} & \cdots & \rho_m \end{bmatrix}, \tag{3.20}$$

$$r = \begin{bmatrix} \rho_{m+1} \\ \cdot \\ \cdot \\ \cdot \\ \rho_{m+n} \end{bmatrix}. \tag{3.21}$$

Sistemul de ecuaţii (3.11)–(3.13) pentru determinarea parametrilor unui semnal AR din măsurători perturbate este un caz particular al sistemului (3.13),(3.20) pentru determinarea parametrilor AR ai unui semnal ARMA(n, m) (într-adevăr, definiţiile (3.11),(3.12) se obţin din (3.20) pentru $m = n$). În continuare vom considera sistemul mai general de ecuaţii (3.13),(3.20). În unele aplicaţii, este necesară rezolvarea acestui sistem pentru $n = 1, 2, \ldots, n_{max}$.

Scanned with OKEN Scanner

Utilizarea în acest scop a unui algoritm standard ar necesita un număr de operaţii aritmetice de ordinul lui n_{max}^4: $\mathcal{O}(n_{max}^4)$. Acest număr de operaţii poate fi redus drastic dacă exploatăm structura specială a matricei R care apare în (3.13): R este o matrice *Toeplitz nesimetrică*. Observând acest fapt şi exploatând şi forma specifică a membrului drept r al sistemului (3.13), putem reduce numărul de operaţii necesare rezolvării lui (3.13) pentru $n = 1, \ldots, n_{max}$, de la $\mathcal{O}(n_{max}^4)$ la $\mathcal{O}(n_{max}^2)$. Pentru valori relativ modeste ale lui n_{max} de ordinul zecilor (valori care pot apare frecvent în aplicaţii), aceasta înseamnă o reducere a timpului de calcul de sute de ori (un avantaj de loc de neglijat pentru majoritatea aplicaţiilor).

Paragraful următor descrie în detaliu modul în care structura specială a ecuaţiilor (3.13), (3.20) poate fi exploatată pentru a obţine un algoritm de tip Levinson de rezolvare rapidă a acestor ecuaţii.

3.2. Algoritm de tip Levinson

Să ataşăm cantităţilor ce apar în (3.13) un indice n pentru a sublinia faptul că ele corespund unui model ARMA a cărui parte AR are ordin n. Astfel, (3.13) devine:

$$R_n \theta_n = -r_n.$$

Să demonstrăm mai întâi un număr de rezultate ajutătoare. Primul este o extensie a Lemei 1.2.1. referitoare la o proprietate simplă dar esenţială a operaţiei de reversare " ˜ " (vezi Capitolul 1).

Lema 3.2.1. *Fie R o matrice Toeplitz generală (posibil nesimetrică) de dimensiune $n \times n$, şi fie*

$$y = Rx. \tag{3.22}$$

Atunci

$$\tilde{y} = R^{\mathrm{T}} \tilde{x}. \tag{3.23}$$

Demonstraţie: Din (3.22) urmează că elementul i al vectorului $\tilde{y}$ are următoarea expresie:

$$\begin{aligned} \tilde{y}_i \;\triangleq\; y_{n-i+1} &= \sum_{k=1}^{n} R_{n-i+1,k} x_k = \sum_{k=1}^{n} \rho_{n-i+1-k} x_k = \\ &= \sum_{p=1}^{n} \rho_{p-i} x_{n-p+1} = \sum_{p=1}^{n} R_{p,i} \tilde{x}_p \sum_{p=1}^{n} (R^{\mathrm{T}})_{i,p} \tilde{x}_p = (R^{\mathrm{T}} \tilde{x})_i, \end{aligned} \tag{3.24}$$

unde, prin definiţie, $R_{i,j} = \rho_{i-j}$. Egalitatea (3.23) urmează imediat din (3.24).

□

Al doilea rezultat preliminar are ca subiect inversarea matricelor partiţionate[76].

Lema 3.2.2. *Să considerăm următoarea matrice partiţionată*

$$S = \begin{bmatrix} A & B \\ D & C \end{bmatrix} \tag{3.25}$$

unde blocurile A şi C sunt pătrate. Să presupunem că matricele A şi $C - DA^{-1}B$ sunt nesingulare. Atunci:

$$S^{-1} = \begin{bmatrix} I \\ 0 \end{bmatrix} A^{-1} \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -A^{-1}B \\ I \end{bmatrix} (C - DA^{-1}B)^{-1} \begin{bmatrix} -DA^{-1} & I \end{bmatrix}. \tag{3.26}$$

Scanned with OKEN Scanner

Dacă C şi $(A - BC^{-1}D)$ sunt matrice nesingulare, atunci matricea inversă S^{-1} poate fi pusă sub următoarea formă:

$$S^{-1} = \begin{bmatrix} 0 \\ I \end{bmatrix} C^{-1} \begin{bmatrix} 0 & I \end{bmatrix} + \begin{bmatrix} I \\ -C^{-1}D \end{bmatrix} (A - BC^{-1}D)^{-1} \begin{bmatrix} I & -BC^{-1} \end{bmatrix}. \tag{3.27}$$

Demonstraţie: Calcule simple arată că

$$\begin{bmatrix} A & B \\ D & C \end{bmatrix} \left\{ \begin{bmatrix} I \\ 0 \end{bmatrix} A^{-1} \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -A^{-1}B \\ I \end{bmatrix} (C - DA^{-1}B)^{-1} \begin{bmatrix} -DA^{-1} & I \end{bmatrix} \right\} =$$
$$= \begin{bmatrix} I & 0 \\ DA^{-1} & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ C - DA^{-1}B \end{bmatrix} (C - DA^{-1}B)^{-1} \begin{bmatrix} -DA^{-1} & I \end{bmatrix} =$$
$$= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{3.28}$$

şi deci forma (3.26) a inversei este demonstrată. Forma (3.27) se demonstrează similar.
□

În continuare să observăm următoarele structuri partiţionate ale matricei R_{n+1} şi vectorului r_{n+1}:

$$R_{n+1} = \begin{bmatrix} R_n & \beta_n \\ \tilde{r}_n^{\mathrm{T}} & \rho_m \end{bmatrix} = \begin{bmatrix} \rho_m & \tilde{\beta}_n^{\mathrm{T}} \\ r_n & R_n \end{bmatrix}, \tag{3.29}$$

$$r_{n+1} = \begin{bmatrix} r_n \\ \rho_{m+n+1} \end{bmatrix}, \tag{3.30}$$

unde

$$\beta_n = \begin{bmatrix} \rho_{m-n} \\ \rho_{m-1} \end{bmatrix}. \tag{3.31}$$

Să observăm, de asemenea, că vectorul β_{n+1} admite următoarea partiţionare:

$$\beta_{n+1} = \begin{bmatrix} \rho_{m-n-1} \\ \beta_n \end{bmatrix}. \tag{3.32}$$

Pentru calculul recursiv–în–ordin al vectorilor $\{\theta_n\}$ sunt necesare un număr de variabile auxiliare. Aceste variabile sunt definite în continuare:

$$\begin{aligned} \Delta_n &= -R_n^{-1}\beta_n, \\ \alpha_n &= \rho_{m+n+1} - \tilde{r}_n^{\mathrm{T}} R_n^{-1} r_n = \rho_{m+n+1} + \tilde{r}_n^{\mathrm{T}}\theta_n, \\ \sigma_n &= \rho_m - \tilde{r}_n^{\mathrm{T}} R_n^{-1}\beta_n = \rho_m + \tilde{r}_n^{\mathrm{T}}\Delta_n, \\ \mu_n &= \rho_{m-n-1} - \tilde{\beta}_n^{\mathrm{T}} R_n^{-1}\beta_n = \rho_{m-n-1} + \tilde{\beta}_n^{\mathrm{T}}\Delta_n. \end{aligned} \tag{3.33}$$

Suntem acum pregătiţi pentru a prezenta o deducere simplă a algoritmului rapid Levinson de rezolvare recursivă–în–ordin a sistemului de ecuaţii liniare (3.13),(3.20).

Scanned with OKEN Scanner

Teorema 3.2.1. *Soluţiile de ordinul n şi, respectiv, $n+1$ ale sistemului (3.13) sunt legate prin următoarele relaţii:*

$$\theta_{n+1} = \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} - \frac{\alpha_n}{\sigma_n} \begin{bmatrix} \Delta_n \\ 1 \end{bmatrix}, \tag{3.34}$$

unde $\{\Delta_n\}$ şi $\{\sigma_n\}$ satisfac următoarele ecuaţii recurente,

$$\Delta_{n+1} = \begin{bmatrix} 0 \\ \Delta_n \end{bmatrix} - \frac{\mu_n}{\sigma_n} \begin{bmatrix} 1 \\ \theta_n \end{bmatrix}, \tag{3.35}$$

$$\sigma_{n+1} = \sigma_n - \alpha_n \mu_n / \sigma_n \tag{3.36}$$

iar α_n şi μ_n sunt scalarii definiţi în (3.33).

Demonstraţie: Utilizând Lema 3.2.2., partiţionările (3.29)–(3.32) şi definiţiile (3.33), putem scrie

$$\begin{aligned} \theta_{n+1} &= -R_{n+1}^{-1} r_{n+1} = -\left\{ \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1} \begin{bmatrix} I & 0 \end{bmatrix} + \right. \\ &+ \left. \begin{bmatrix} -R_n^{-1}\beta_n \\ 1 \end{bmatrix} \begin{bmatrix} -\tilde{r}_n^{\mathrm{T}} R_n^{-1} & 1 \end{bmatrix} / (\rho_m - \tilde{r}_n^{\mathrm{T}} R_n^{-1} \beta_n) \right\} \begin{bmatrix} r_n \\ \rho_{m+n+1} \end{bmatrix} = \\ &= \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} - \frac{\alpha_n}{\sigma_n} \begin{bmatrix} \Delta_n \\ 1 \end{bmatrix} \end{aligned} \tag{3.37}$$

şi, în mod similar,

$$\begin{aligned} \Delta_{n+1} &= -R_{n+1}^{-1} \beta_{n+1} = -\left\{ \begin{bmatrix} 0 \\ I \end{bmatrix} R_n^{-1} \begin{bmatrix} 0 & I \end{bmatrix} + \right. \\ &+ \left. \begin{bmatrix} 1 \\ -R_n^{-1} r_n \end{bmatrix} \begin{bmatrix} 1 & -\tilde{\beta}_n^{\mathrm{T}} R_n^{-1} \end{bmatrix} / (\rho_m - \tilde{\beta}_n^{\mathrm{T}} R_n^{-1} r_n) \right\} \begin{bmatrix} \rho_{m-n-1} \\ \beta_n \end{bmatrix} = \\ &= \begin{bmatrix} 0 \\ \Delta_n \end{bmatrix} - \frac{\mu_n}{\bar{\sigma}_n} \begin{bmatrix} 1 \\ \theta_n \end{bmatrix}, \end{aligned} \tag{3.38}$$

unde

$$\bar{\sigma}_n = \rho_m - \tilde{\beta}_n^{\mathrm{T}} R_n^{-1} r_n. \tag{3.39}$$

Ecuaţia recurentă (3.34) este, astfel, demonstrată. Pentru a demonstra şi (3.35) trebuie să arătăm că $\bar{\sigma}_n = \sigma_n$. Totuşi această egalitate se verifică imediat utilizând Lema 3.2.1.:

$$\bar{\sigma}_n = \rho_m - r_n^{\mathrm{T}} R_n^{-\mathrm{T}} \tilde{\beta}_n = \rho_m + r_n^{\mathrm{T}} \tilde{\Delta}_n = \rho_m + \tilde{r}_n^{\mathrm{T}} \Delta_n = \sigma_n. \tag{3.40}$$

Mai rămâne să verificăm relaţia recursivă (3.36) pentru calculul lui $\{\sigma_n\}$. Din ecuaţia de definiţie a lui σ_n şi (3.35), obţinem

$$\begin{aligned} \sigma_{n+1} &= \rho_m + \tilde{r}_{n+1}^{\mathrm{T}} \Delta_{n+1} = \rho_m + \begin{bmatrix} \rho_{m+n+1} & \tilde{r}_n^{\mathrm{T}} \end{bmatrix} \left\{ \begin{bmatrix} 0 \\ \Delta_n \end{bmatrix} - -\frac{\mu_n}{\sigma_n} \begin{bmatrix} 1 \\ \theta_n \end{bmatrix} \right\} = \\ &= \rho_m + \tilde{r}_n^{\mathrm{T}} \Delta_n - \frac{\mu_n}{\sigma_n} (\rho_{m+n+1} + \tilde{r}_n^{\mathrm{T}} \theta_n) = \sigma_n - \mu_n \alpha_n / \sigma_n \end{aligned} \tag{3.41}$$

Scanned with OKEN Scanner

Ecuaţie	Număr operaţii [flop]
Iniţializare:	
$\theta_1 = -\frac{\rho_{m+1}}{\rho_m}$	$\mathcal{O}(1)$
$\Delta_1 = -\frac{\rho_{m-1}}{\rho_m}$	$\mathcal{O}(1)$
$\sigma_1 = \rho_m(1-\theta_1\Delta_1)$	$\mathcal{O}(1)$
Pentru $n = 1,\ldots,n_{max}$ calculează:	
$\alpha_n = \rho_{m+n+1} + \tilde{r}_n^{\mathrm{T}}\theta_n$	$\sim n$
$\mu_n = \rho_{m-n-1} + \tilde{\beta}_n^{\mathrm{T}}\Delta_n$	$\sim n$
$\theta_{n+1} = \begin{bmatrix}\theta_n\\0\end{bmatrix} - \frac{\alpha_n}{\sigma_n}\begin{bmatrix}\Delta_n\\1\end{bmatrix}$	$\sim n$
$\Delta_{n+1} = \begin{bmatrix}0\\\Delta_n\end{bmatrix} - \frac{\mu_n}{\sigma_n}\begin{bmatrix}1\\\theta_n\end{bmatrix}$	$\sim n$
$\sigma_{n+1} = \sigma_n - \mu_n\alpha_n/\sigma_n$	$\mathcal{O}(1)$

Tabelul 3.1. Algoritmul Levinson pentru sisteme Toeplitz asimetrice.

şi demonstraţia este încheiată.
□

Ecuaţiile recursive (3.34)–(3.36) trebuie iniţializate cu θ_1, Δ_1 şi σ_1. Expresii pentru aceste cantităţi iniţiale se obţin uşor din relaţiile lor de definiţie:

$$\begin{aligned}\theta_1 &= -\rho_{m+1}/\rho_m,\\ \Delta_1 &= -\rho_{m-1}/\rho_m,\\ \sigma_1 &= \rho_m(1-\theta_1\Delta_1).\end{aligned} \tag{3.42}$$

Relaţiile recurente (3.34)–(3.36) împreună cu definiţiile (3.33) ale lui α_n şi μ_n şi cu iniţializările (3.42) constituie algoritmul rapid Levinson pentru sisteme Toeplitz asimetrice de tipul (3.13), (3.20). Pentru convenienţa cititorului, ecuaţiile acestui algoritm sunt prezentate în Tabelul 3.1., împreună cu numărul de operaţii aritmetice aferente. Observăm din aceast tabel că numărul de "flopi" (vezi definiţia (1.32)) necesar AL (algoritmului Levinson) pentru a trece de la modelul de ordin n la cel de ordin $n+1$ este aproximativ egal cu $4n$. Astfel, numărul total de [flopi] necesar AL pentru a calcula parametrii AR $\{\theta_n\}$ ai tuturor modelelor de ordine $n = 1, 2, \ldots, n_{max}$ este, în mod aproximativ, următorul:

$$4\sum_{n=1}^{n_{max}} n \approx 2n_{max}^2 \ \text{[flop]}. \tag{3.43}$$

Asfel AL este, într-adevăr, cu două ordine de mărime mai rapid decât un algoritm standard utilizat pentru a calcula $\{\theta_n\}_{n=1,\ldots,n_{max}}$. Comparând (1.33) şi (3.43) observăm că numărul de operaţii necesare unui AL pentru a rezolva un sistem Toeplitz asimetric este de circa 2 ori mai mare decât cel necesar pentru rezolvarea unui sistem Toeplitz simetric (desigur, era de aşteptat ca AL pentru cazul asimetric să fie mai complex din punct de vedere numeric).

Este interesant de observat că AL din Tabelul 3.1. lucrează "normal" dacă şi numai dacă

$$\sigma_n \neq 0 \ \ (n = 1, 2, \ldots) \text{ şi } \rho_m \neq 0 \tag{3.44}$$

Scanned with OKEN Scanner

(dacă $\rho_m = 0$ atunci AL trebuie stopat chiar în faza de iniţializare ; dacă $\sigma_n = 0$ atunci AL trebuie stopat la iteraţia $n+1$). Pe de altă parte, este evident că diversele cantităţi care apar în AL sunt definite dacă şi numai dacă

$$\det R_n \neq 0, \quad n = 1, 2, \ldots \tag{3.45}$$

Condiţiile (3.44) şi (3.45) ar trebui să fie echivalente. De fapt, *echivalenţa condiţiilor* (3.44) şi (3.45) este uşor de arătat. Să considerăm mai întâi implicaţia (3.45) $\Rightarrow$ (3.44). Când (3.45) este satisfăcută, soluţiile $\{\theta_n\}_{n=1,2,\ldots}$ ale sistemelor (3.13) sunt unic definite şi deci putem scrie

$$R_{n+1}\begin{bmatrix} 1 & 0 \\ \theta_n & I \end{bmatrix} = \begin{bmatrix} \rho_m & \tilde{\beta}_n^{\mathrm{T}} \\ r_n & R_n \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \theta_n & I \end{bmatrix} = \begin{bmatrix} \sigma_n & \tilde{\beta}_n^{\mathrm{T}} \\ 0 & R_n \end{bmatrix}. \tag{3.46}$$

Din (3.46) obţinem următoarea implicaţie

$$\det R_{n+1} \neq 0 \quad \Rightarrow \quad \{\sigma_n \neq 0 \text{ şi } \det R_n \neq 0\} \tag{3.47}$$

din care conchidem uşor, prin inducţie, că (3.45) $\Rightarrow$ (3.44).

Pentru a demonstra implicaţia (3.44) $\Rightarrow$ (3.45), să observăm că dacă (3.44) este satisfăcută, atunci θ_1 există. Astfel relaţia (3.46) este validă pentru $n = 1$, fapt care împreună cu $\sigma_1 \neq 0$ implică $\det R_2 \neq 0$. Astfel θ_2 există ş.a.m.d.

Inainte de a încheia acest paragraf, să mai observăm următorul fapt: condiţia (3.45) (sau, în mod echivalent, (3.44)) este necesară şi suficientă pentru existenţa tuturor soluţiilor $\{\theta_n\}_{n=1,2,\ldots}$. Totuşi, dacă se doreşte determinarea lui θ_M, pentru un M dat, şi nu a tuturor vectorilor $\{\theta_1, \ldots, \theta_M\}$, atunci (3.44) este o condiţie suficientă dar nu şi necesară. În acest ultim caz este posibil ca AL să nu poată determina θ_M, în pofida faptului că θ_M există şi este unic definit. Desigur, se poate obiecta că în aplicaţii $\{\sigma_n\}$ nu va fi exact egal cu zero chiar dacă matricea R_M are anumite secţiuni singulare (motivul fiind erorile de rotunjire ce afectează calculele în virgulă mobilă). Astfel, pe un calculator numeric ecuaţiile recurente ale AL vor putea fi iterate până la $n = M$ (deci θ_M va fi calculat) chiar dacă, teoretic, anumite valori ale secvenţei $\{\sigma_n\}$ erau zero. Totuşi, ne putem aştepta ca într-un astfel de caz, rezultatul final furnizat de AL să fie puternic afectat de erori numerice şi, deci, inutilizabil.

3.3. Paralelisme între AL şi ALD

Pentru $m = 0$, AL se reduce la ALD pentru cazul Toeplitz simetric, aşa cum era de aşteptat.Pentru a verifica acest lucru, observaţi că pentru $m = 0$ matricea R_n este simetrică şi deci $\beta_n = \tilde{r}_n$. Astfel $\Delta_n = \tilde{\theta}_n$ şi $\mu_n = \alpha_n$. Prin urmare, ecuaţiile pentru aceste două variabile din Tabelul 3.1. sunt redundante şi, deci, pot fi eliminate. Ecuaţiile care rămân sunt în mod precis cele ale ALD.

Pentru $m \neq 0$, AL este mai complex decât ALD aşa cum (din nou) era de aşteptat şi cum, de altfel, am subliniat deja în paragraful anterior. Pentru AL efortul de calcul este de două ori mai mare decât cel asociat cu ALD, datorită necesităţii de a calcula două variabile scalare:

$$\begin{aligned} \overline{k}_n &= -\alpha_n/\sigma_n, \\ \tilde{k}_n &= -\mu_n/\sigma_n \end{aligned} \tag{3.48}$$

(în locul uneia singure, k_n, pentru ALD) şi de a adapta coeficienţii a două polinoame

$$\begin{aligned} A_n(z) &= 1 + \begin{bmatrix} z & \dots & z^n \end{bmatrix} \theta_n, \\ B_n(z) &= \begin{bmatrix} 1 & z & \dots & z^{n-1} \end{bmatrix} \Delta_n + z^n \end{aligned} \tag{3.49}$$

(în locul unuia singur, $\{A_n(z)\}$, pentru ALD).

Definiţiile introduse mai sus pot fi utilizate pentru a scrie AL sub o formă compactă comparabilă cu ecuaţia (1.77) corespunzătoare ALD. Într-adevăr înmulţind la stânga ecuaţia (3.34) cu $\begin{bmatrix} z & \dots & z^{n+1} \end{bmatrix}$ şi (3.35) cu $\begin{bmatrix} 1 & z & \dots & z^n \end{bmatrix}$ şi adăugând, respectiv, 1 şi z^{n+1} la ambii membri ai ecuaţiilor rezultate, obţinem următoarele identităţi polinomiale:

$$\begin{aligned} A_{n+1}(z) &= A_n(z) + \overline{k}_n z B_n(z), \\ B_{n+1}(z) &= z B_n(z) + \tilde{k}_n A_n(z), \end{aligned} \tag{3.50}$$

sau, echivalent,

$$\begin{bmatrix} A_{n+1}(z) \\ B_{n+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & \overline{k}_n z \\ \tilde{k}_n & z \end{bmatrix} \begin{bmatrix} A_n(z) \\ B_n(z) \end{bmatrix}. \tag{3.51}$$

Pentru $m = 0, \overline{k}_n = \tilde{k}_n = k_n$ şi $B_n(z) = A_n^*(z)$ (polinomul reciproc lui $A_n(z)$) şi, prin urmare, (3.51) se reduce la (1.77). În cazul general corespunzător lui $m \neq 0$, ecuaţia (3.51) este mai complexă decât (1.77), dar analogia dintre cele două ecuaţii rămâne puternică. Această analogie ne face să ne întrebăm dacă AL nu are proprietăţi similare cu cele ale ALD referitoare la stabilitatea polinoamelor $\{A_n(z)\}$. Din păcate se poate arăta că AL nu posedă astfel de proprietăţi: polinoamele $\{A_n(z)\}$ determinate cu AL *nu* sunt în mod necesar stabile; mai mult, informaţii despre localizarea rădăcinilor lui $A_n(z)$ în raport cu cercul unitar *nu* se pot obţine comparând elementele secvenţei $\{|\overline{k}_n|, |\tilde{k}_n|\}$ cu unu.

Continuând studiul paralel al AL şi ALD, ne putem întreba dacă AL poate fi utilizat pentru a determina o factorizare a matricei inverse de covarianţă R_n^{-1}. Pentru a realiza că această posibilitate există în mod real, să observăm că din ecuaţia (3.46) putem obţine cu uşurinţă următoarea identitate:

$$\begin{bmatrix} 1 & \tilde{\Delta}_n^{\mathrm{T}} \\ 0 & I \end{bmatrix} R_{n+1} \begin{bmatrix} 1 & 0 \\ \theta_n & I \end{bmatrix} = \begin{bmatrix} \sigma_n & 0 \\ 0 & R_n \end{bmatrix}. \tag{3.52}$$

Înlocuind pe n cu $n-1$ în (3.52), obţinem

$$\begin{bmatrix} 1 & \tilde{\Delta}_{n-1}^{\mathrm{T}} \\ 0 & I \end{bmatrix} R_n \begin{bmatrix} 1 & 0 \\ \theta_{n-1} & I \end{bmatrix} = \begin{bmatrix} \sigma_{n-1} & 0 \\ 0 & R_{n-1} \end{bmatrix}. \tag{3.53}$$

Utilizând ambele ecuaţii anterioare, putem scrie:

$$\begin{bmatrix} 1 & & \tilde{\Delta}_n^{\mathrm{T}} \\ 0 & 1 & \tilde{\Delta}_{n-1}^{\mathrm{T}} \\ 0 & 0 & I \end{bmatrix} R_{n+1} \begin{bmatrix} 1 & 0 & 0 \\ \theta_n & 1 & 0 \\ 0 & \theta_{n-1} & I \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \tilde{\Delta}_{n-1}^{\mathrm{T}} \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} 1 & \tilde{\Delta}_n^{\mathrm{T}} \\ 0 & I \end{bmatrix} R_{n+1} \begin{bmatrix} 1 & 0 \\ \theta_n & I \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \theta_{n-1} & & I \end{bmatrix} =$$

Scanned with OKEN Scanner

$$= \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & \tilde{\Delta}_{n-1}^{\mathrm{T}} \\ 0 & 0 & I \end{array}\right] \left[\begin{array}{cc} \sigma_n & 0 \\ 0 & R_n \end{array}\right] \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \theta_{n-1} & I & \end{array}\right] \left[\begin{array}{ccc} \sigma_n & 0 & 0 \\ 0 & \sigma_{n-1} & 0 \\ 0 & 0 & R_{n-1} \end{array}\right]. \tag{3.54}$$

Continuând în aceeaşi manieră obţinem în final următoarea ecuaţie:

$$\begin{bmatrix} 1 & & \tilde{\Delta}_n^{\mathrm{T}} \\ 0 & 1 & \tilde{\Delta}_{n-1}^{\mathrm{T}} \\ & \ddots & \\ & 1 & \tilde{\Delta}_1^{\mathrm{T}} \\ 0 & & 1 \end{bmatrix} R_{n+1} \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ & 1 & \cdots & 0 & 0 \\ \theta_n & & \cdots & 0 & 0 \\ & \theta_{n-1} & \cdots & 1 & 0 \\ & & \cdots & \theta_1 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_n & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{n-1} & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & \sigma_1 & 0 \\ 0 & 0 & \cdots & 0 & \rho_m \end{bmatrix} \tag{3.55}$$

sau, utilizând notaţii evidente,

$$V_{n+1}^{\mathrm{T}} R_{n+1} U_{n+1} = D_{n+1}. \tag{3.56}$$

Din (3.56) se obţine imediat următoarea factorizare inferior–superior triunghiluară a lui R_{n+1}^{-1}

$$R_{n+1}^{-1} = U_{n+1} D_{n+1} V_{n+1}^{\mathrm{T}}, \tag{3.57}$$

care este similară factorizării (1.71) furnizată de ALD. În comparaţie cu factorizarea Cholesky (1.71) validă în cazul simetric, factorizarea asimetrică (3.57) are, totuşi, mai puţine implicaţii teoretice şi, utilizări practice.

Scanned with OKEN Scanner

Capitolul 4

Algoritmul Whittle–Wiggins–Robinson pentru Modele AR Multivariabile

În acest capitol, rezultatele referitoare la modele AR monovariabile prezentate în Capitolul 1 vor fi extinse la semnale multivariabile.

4.1. Modele AR multivariabile

Fie $\{y(t)\}_{t=1,2,\ldots}$ un semnal m–dimensional. Să considerăm următorul model AR pentru $y(t)$ (model care reprezintă o extensie directă a modelului AR scalar (1.1)):

$$y(t) = -A_1 y(t-1) - \ldots - A_n y(t-n) + \varepsilon(t), \tag{4.1}$$

unde $\{A_i\}$ sunt matrice $m \times m$, iar $\varepsilon(t)$ este un vector m–dimensional. Ecuaţia (4.1) se mai numeşte şi *model de predicţie liniară*, pentru un motiv destul de transparent: primii n termeni în (4.1) formează o predicţie liniară a lui $y(t)$, calculată pe baza observaţiilor imediat precedente $\{y(t-1), \ldots, y(t-n)\}$ ale acestei variabile, iar ultimul termen $\varepsilon(t)$ poate fi interpretat ca *eroare de predicţie.*

Problema principală în practică este determinarea coeficienţilor matriceali $\{A_i\}$ din măsurători ale semnalului $\{y(t)\}$. În majoritatea aplicaţiilor se doreşte şi determinarea matricei de covarianţă a erorilor de predicţie (sau "reziduurilor") $\varepsilon(t)$,

$$Q = E\varepsilon(t)\varepsilon^{\mathrm{T}}(t). \tag{4.2}$$

Metoda cea mai frecvent utilizată în acest scop este aceea a CMMP extinsă la cazul multivariabil: parametrii $\{A_i\}$ se determină prin minimizarea lui (4.2), iar Q se ia egal cu valoarea lui (4.2) în punctul de minim. Să notăm că (4.2) este un *criteriu matriceal*; minimizarea lui în raport cu $\{A_i\}$ se face în conformitate cu relaţiile de ordine definite pe domeniul matricelor pozitiv (semi)definite: dacă A şi B sunt două matrice pozitive (semi) definite, atunci scriem $A \geq B$ ($A > B$) dacă matricea diferenţă $A - B$ este pozitiv semidefinită (pozitiv definită) (în particular, $A \geq 0$ ($A > 0$) înseamnă că A este o matrice pozitiv semidefinită (pozitiv definită)). Să mai notăm că metoda CMMP, descrisă succint mai sus, determină "cel mai bun predictor liniar de ordin n" (în sensul că orice alt predictor liniar, de acelaşi ordin, va avea erori de predicţie cu o

Scanned with OKEN Scanner

matrice de covarianţă mai mare decât aceea corespunzătoare modelului CMMP). Totuşi modelul AR furnizat de metoda CMMP este util nu numai pentru *predicţie*, ci şi pentru multe alte aplicaţii: *analiză spectrală*, *prelucrarea imaginilor*, *compresia datelor* etc. [16],[23],[27],[45],[48].

Să considerăm în continuare problema minimizării lui Q în raport cu $\{A_i\}$. Să introducem următoarele notaţii:

$$R_k = Ey(t)y^{\mathrm{T}}(t-k) = R_{-k}^{\mathrm{T}}, \quad k = 0, 1, 2, \ldots \tag{4.3}$$

$$\begin{aligned} \Gamma &= E\begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} \begin{bmatrix} y^{\mathrm{T}}(t-1) & \ldots & y^{\mathrm{T}}(t-n) \end{bmatrix} = \\ &= \begin{bmatrix} R_0 & R_1 & \ldots & R_{n-1} \\ R_{-1} & R_0 & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ R_{-(n-1)} & & & R_0 \end{bmatrix}, \end{aligned} \tag{4.4}$$

$$\gamma = Ey(t)\begin{bmatrix} y^{\mathrm{T}}(t-1) & \ldots & y^{\mathrm{T}}(t-n) \end{bmatrix} = \begin{bmatrix} R_1 & \ldots & R_n \end{bmatrix}, \tag{4.5}$$

$$\theta = \begin{bmatrix} A_1 \ldots A_n \end{bmatrix} \tag{4.6}$$

(observaţi că, în acest caz, θ nu mai este un vector n-dimensional ca în cazul monovariabil, ci o matrice $m \times (m \cdot n)$). Utilizând notaţiile anterioare, putem scrie R sub următoarea formă:

$$\begin{aligned} Q &= E\left\{ y(t) + \theta \begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} \right\} \left\{ y^{\mathrm{T}}(t) + \begin{bmatrix} y^{\mathrm{T}}(t-1) & \ldots & y^{\mathrm{T}}(t-n) \end{bmatrix} \theta^{\mathrm{T}} \right\} = \\ &= R_0 + \gamma\theta^{\mathrm{T}} + \theta\gamma^{\mathrm{T}} + \theta\Gamma\theta^{\mathrm{T}} = \\ &= (\theta + \gamma\Gamma^{-1})\Gamma(\theta + \gamma\Gamma^{-1})^{\mathrm{T}} + R_0 - \gamma\Gamma^{-1}\gamma^{\mathrm{T}}. \end{aligned} \tag{4.7}$$

Matricea Γ este cel puţin pozitiv semidefinită (vezi (1.12)). Am presupus implicit în (4.7) că Γ este nesingulară. Astfel, $\Gamma > 0$; ca o consecinţă singurul termen din (4.7) care depinde de θ este o matrice pozitiv semidefinită. Rezultă, deci, imediat că Q este minimizată pentru

$$\theta = -\gamma\Gamma^{-1}, \tag{4.8}$$

valoarea minimă corespunzătoare fiind

$$Q = R_0 - \gamma\Gamma^{-1}\gamma^{\mathrm{T}} = R_0 + \theta\gamma^{\mathrm{T}}. \tag{4.9}$$

În aplicaţii practice, covarianţele teoretice care apar în (4.8) şi (4.9) vor fi înlocuite prin covarianţe de eşantion calculate, de exemplu, cu următoarea formulă:

$$R_k = \frac{1}{N}\sum_{t=k+1}^{N} y(t)y^{\mathrm{T}}(t-k) = R_{-k}^{\mathrm{T}} \quad (k = 0, 1, 2, \ldots), \tag{4.10}$$

Scanned with OKEN Scanner

unde N este numărul de măsurători disponibile. Similar cazului monovariabil, nu vom introduce o notaţie specială pentru matricele Γ, γ etc. construite din covarianţe de eşantion, pentru a nu complica inutil scrierea.

În mod frecvent, în aplicaţii, este necesară determinarea lui θ şi Q pentru toate ordinele n de la 1 până la un ordin maxim, să zicem n_{max}.

Utilizarea unui algoritm standard în acest scop ar necesita $\mathcal{O}(m^3 n_{max}^4)$ operaţii aritmetice. Pentru anumite aplicaţii acest număr de operaţii poate fi intolerabil de mare. Problema reducerii lui este, evident, mai pregnantă decât în cazul monovariabil unde $m = 1$.

Paragraful următor descrie în detaliu un algoritm rapid pentru determinarea lui θ şi Q date de ecuaţiile (4.8) şi (4.9). Acest algoritm exploatează structura specială a matricei Γ, pentru a rezolva sistemele liniare de ecuaţii (4.8), corespunzătoare lui $n = 1, \ldots, n_{max}$, în $\mathcal{O}(m^3 n_{max}^2)$ operaţii aritmetice (deci de circa n_{max}^2 ori mai rapid decât un algoritm standard). El va fi numit algoritmul Whittle–Wiggins–Robinson (W^{2}R), după numele celor care l-au introdus în literatura de specialitate[96],[97].

4.2. Algoritmul Whittle–Wiggins–Robinson

Să ataşăm cantităţilor ce apar în (4.8) şi (4.9) un indice n pentru a sublinia faptul că ele corespund modelului de ordin n. Astfel (4.8) şi (4.9) devin, respectiv,

$$\theta_n = -\gamma_n \Gamma_n^{-1}, \tag{4.11}$$

$$Q_n = R_0 + \theta_n \gamma_n^{\mathrm{T}}. \tag{4.12}$$

Cele două ecuaţii matriceale anterioare pot fi combinate într-una singură

$$\begin{bmatrix} I & \theta_n \end{bmatrix} \Gamma_{n+1} = \begin{bmatrix} Q_n & 0 & \ldots & 0 \end{bmatrix} \tag{4.13}$$

cum se poate uşor verifica făcând uz de următoarea partiţionare a matricei Γ_{n+1}:

$$\Gamma_{n+1} = \begin{bmatrix} R_0 & \gamma_n \\ \gamma_n^{\mathrm{T}} & \Gamma_n \end{bmatrix}. \tag{4.14}$$

Matricea Γ_{n+1} care apare în ecuaţiile anterioare *nu* mai este Toeplitz ca în cazul monovariabil. Totuşi chiar pentru $m > 1$, Γ_{n+1} are o structură specială: ea este ceea ce numim o *matrice Toeplitz de blocuri* (adică blocul i, j al lui Γ_{n+1} nu depinde şi de i şi de j, ci numai de diferenţa lor $(i - j)$); în plus Γ_{n+1} este o *matrice simetrică*: $\Gamma_{n+1}^{\mathrm{T}} = \Gamma_{n+1}$. Exploatarea acestei structuri particulare a lui Γ_{n+1} face posibil calculul rapid al $\{\theta_n, Q_n\}_{n=1,2,\ldots}$, aşa cum vom arăta în continuarea acestui paragraf.

Rezolvarea eficientă a sistemului de ecuaţii liniare (4.13) cere introducerea unor variabile auxiliare. Astfel, să definim matricele β_n de dimensiunea $m \times (m \cdot n)$ şi S_n de dimensiune $m \times m$, prin următoarea ecuaţie:

$$\begin{bmatrix} \beta_n & I \end{bmatrix} \Gamma_{n+1} = \begin{bmatrix} 0 & \ldots & 0 & S_n \end{bmatrix}. \tag{4.15}$$

Este interesant de observat că în cazul monovariabil ($m = 1$), caz în care Γ_{n+1} este o matrice Toeplitz simetrică, simpla invocare a Lemei 1.2.1. şi a unicităţii soluţiei sistemului (4.15) arată imediat că $\beta_n = \tilde{\theta}_n$ şi $S_n = Q_n$! În cazul multivariabil ($m > 1$), Lema 1.2.1. nu mai este

Scanned with OKEN Scanner

aplicabilă, iar egalităţile anterioare nu mai sunt valide, în general. Acesta este, în esenţă, motivul pentru care algoritmul multivariabil W²R are o structură puţin mai complicată decât omologul său scalar ALD.

Să mai introducem următoarea matrice de dimensiune $m \times m$:

$$P_n = R_{n+1} + \theta_n \begin{bmatrix} R_n \\ \cdot \\ \cdot \\ R_1 \end{bmatrix} \tag{4.16}$$

şi să observăm că

$$P_n = R_{n+1} - \begin{bmatrix} R_1 & \dots & R_n \end{bmatrix} \Gamma_n^{-1} \begin{bmatrix} R_n \\ \cdot \\ \cdot \\ R_1 \end{bmatrix} =$$

$$= \left\{ R_{-(n+1)} - \begin{bmatrix} R_{-n} & \dots & R_{-1} \end{bmatrix} \Gamma_n^{-1} \begin{bmatrix} R_{-1} \\ \cdot \\ \cdot \\ R_{-n} \end{bmatrix} \right\}^{\mathrm{T}} =$$

$$= \left\{ R_{-(n+1)} + \beta_n \begin{bmatrix} R_{-1} \\ \cdot \\ \cdot \\ R_{-n} \end{bmatrix} \right\}^{\mathrm{T}}. \tag{4.17}$$

În stabilirea ultimei egalităţi în (4.17) am utilizat (4.15) şi următoarea partiţionare a lui Γ_{n+1},

$$\Gamma_{n+1} = \left[\begin{array}{ccc|c} & \Gamma_n & & \begin{matrix} R_n \\ \cdot \\ \cdot \\ R_1 \end{matrix} \\ \hline R_{-n} & \dots & R_{-1} & R_0 \end{array}\right], \tag{4.18}$$

care împreună implică

$$\beta_n \Gamma_n + \begin{bmatrix} R_{-n} & \dots & R_{-1} \end{bmatrix} = 0 \tag{4.19}$$

sau, echivalent,

$$\beta_n = - \begin{bmatrix} R_{-n} & \dots & R_{-1} \end{bmatrix} \Gamma_n^{-1}. \tag{4.20}$$

Din (4.17) rezultă că matricea P_n poate fi exprimată şi ca funcţie de β_n:

$$P_n^{\mathrm{T}} = R_{-(n+1)} + \beta_n \begin{bmatrix} R_{-1} \\ \cdot \\ \cdot \\ R_{-n} \end{bmatrix}, \tag{4.21}$$

Scanned with OKEN Scanner

Suntem acum pregătiţi pentru a deduce ecuaţiile recurente de bază ale algoritmului W²R.

Teorema 4.2.1. *Soluţiile de ordin n şi $n+1$ ale sistemului liniar de ecuaţii (4.13) sunt legate prin următoarele relaţii:*

$$\theta_{n+1} = \begin{bmatrix} \theta_n & 0 \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} \beta_n & I \end{bmatrix}, \tag{4.22}$$

$$\beta_{n+1} = \tilde{K}_{n+1}^{\mathrm{T}} \begin{bmatrix} I & \theta_n \end{bmatrix} + \begin{bmatrix} 0 & \beta_n \end{bmatrix}, \tag{4.23}$$

$$Q_{n+1} = Q_n + \overline{K}_{n+1} P_n^{\mathrm{T}}, \tag{4.24}$$

$$S_{n+1} = S_n + \tilde{K}_{n+1}^{\mathrm{T}} P_n, \tag{4.25}$$

unde

$$\overline{K}_{n+1} = -P_n S_n^{-1}, \tag{4.26}$$

$$\tilde{K}_{n+1} = -Q_n^{-1} P_n. \tag{4.27}$$

Demonstraţie: În conformitate cu ecuaţiile de definiţie (4.13) şi (4.15), soluţia de ordin $(n+1)$ satisface următoarea ecuaţie matriceală:

$$\begin{bmatrix} I & \theta_{n+1} \\ \beta_{n+1} & I \end{bmatrix} \Gamma_{n+2} = \begin{bmatrix} Q_{n+1} & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & S_{n+1} \end{bmatrix}. \tag{4.28}$$

Combinând (4.13),(4.15),(4.16) şi (4.21) obţinem următoarea ecuaţie

$$\begin{bmatrix} I & \theta_n & 0 \\ 0 & \beta_n & I \end{bmatrix} \Gamma_{n+2} = \begin{bmatrix} Q_n & 0 & \dots & 0 & P_n \\ P_n^{\mathrm{T}} & 0 & \dots & 0 & S_n \end{bmatrix} \tag{4.29}$$

(în stabilirea identităţii anterioare, am făcut uz şi de partiţionările (4.14) şi (4.18) ale lui Γ_{n+2}). Ecuaţia matriceală (4.29) ar avea exact structura lui (4.28) dacă am reuşi să o modificăm astfel încât să introducem zerouri în poziţiile ocupate în (4.29) de P_n şi P_n^{T}. Totuşi, acest deziderat se poate realiza uşor printr-o transformare liniară aplicată ambilor membri ai lui (4.29):

$$\begin{bmatrix} I & \overline{K}_{n+1} \\ \tilde{K}_{n+1}^{\mathrm{T}} & I \end{bmatrix} \begin{bmatrix} I & \theta_n & 0 \\ 0 & \beta_n & I \end{bmatrix} \Gamma_{n+2} = \\ = \begin{bmatrix} I & \overline{K}_{n+1} \\ \tilde{K}_{n+1}^{\mathrm{T}} & I \end{bmatrix} \begin{bmatrix} Q_n & 0 & \dots & 0 & P_n \\ P_n^{\mathrm{T}} & 0 & \dots & 0 & S_n \end{bmatrix}. \tag{4.30}$$

Calcule simple arată că ecuaţia anterioară poate fi scrisă sub următoarea formă,

$$\begin{bmatrix} I & \begin{bmatrix} \theta_n & 0 \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} \beta_n & I \end{bmatrix} \\ \tilde{K}_{n+1}^{\mathrm{T}} \begin{bmatrix} I & \theta_n \end{bmatrix} + \begin{bmatrix} 0 & \beta_n \end{bmatrix} & I \end{bmatrix} \Gamma_{n+2} = \\ = \begin{bmatrix} Q_n + \overline{K}_{n+1} P_n^{\mathrm{T}} & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & S_n + \tilde{K}_{n+1}^{\mathrm{T}} P_n \end{bmatrix}. \tag{4.31}$$

Comparând ecuaţiile (4.28) şi (4.31) şi ţinând cont de faptul că, prin presupunere, soluţia de ordin $n+1$ $(n = 0, 1, \dots)$ este unică, obţinem imediat ecuaţiile recurente (4.22)–(4.25).

□

Scanned with OKEN Scanner

Ecuaţie	Număr operaţii [flop]
Iniţializare:	
$\theta_1 = -R_1 R_0^{-1}$	$\mathcal{O}(m^3)$
$\beta_1 = -R_1^{\mathrm{T}} R_0^{-1}$	$\mathcal{O}(m^3)$
$Q_1 = R_0 + \theta_1 R_1^{\mathrm{T}}$	$\mathcal{O}(m^3)$
$S_1 = R_0 + \beta_1 R_1$	$\mathcal{O}(m^3)$
Pentru $n = 1, \ldots, n_{max}$ calculează:	
$P_n = R_{n+1} + \theta_n \begin{bmatrix} R_n \\ \cdot \\ \cdot \\ \cdot \\ R_1 \end{bmatrix}$	$\sim n\mathcal{O}(m^3)$
$\overline{K}_{n+1} = -P_n S_n^{-1}$	$\mathcal{O}(m^3)$
$\tilde{K}_{n+1} = -Q_n^{-1} P_n$	$\mathcal{O}(m^3)$
$Q_{n+1} = Q_n + \overline{K}_{n+1} P_n^{\mathrm{T}}$	$\mathcal{O}(m^3)$
$S_{n+1} = S_n + \tilde{K}_{n+1}^{\mathrm{T}} P_n$	$\mathcal{O}(m^3)$
$\theta_{n+1} = \begin{bmatrix} \theta_n & 0 \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} \beta_n & I \end{bmatrix}$	$\sim n\mathcal{O}(m^3)$
$\beta_{n+1} = \tilde{K}_{n+1}^{\mathrm{T}} \begin{bmatrix} I & \theta_n \end{bmatrix} + \begin{bmatrix} 0 & \beta_n \end{bmatrix}$	$\sim n\mathcal{O}(m^3)$

Tabelul 4.1. Algoritmul Whittle–Wiggins–Robinson.

Relaţiile recursive (4.22)–(4.25) trebuie iniţializate cu θ_1, β_1, Q_1 şi S_1. Aceste valori iniţiale se obţin din ecuaţiile corespunzătoare de definiţie:

$$\begin{aligned} \theta_1 &= -R_1 R_0^{-1}, \qquad \beta_1 = -R_1^{\mathrm{T}} R_0^{-1}, \\ Q_1 &= R_0 + \theta_1 R_1^{\mathrm{T}}, \qquad S_1 = R_0 + \beta_1 R_1. \end{aligned} \tag{4.32}$$

Tabelul 4.1. prezintă algoritmul W²R într-o manieră sintetică şi indică numărul de operaţii necesare efectuării fiecărui pas al acestui algoritm.

Se observă din tabel că numărul de flopi necesar trecerii de la soluţia de ordin n la cea de ordin $n+1$ este $\sim 3n\mathcal{O}(m^3)$ (presupunând că $n \gg 1$). Rezultă că numărul total de operaţii aritmetice pe care le efectuează algoritmul W²R pentru a calcula $\{\theta_n, Q_n\}(n = 1, \ldots, n_{max})$ este, aproximativ,

$$1,5 n_{max}^2 \mathcal{O}(m^3) \quad [flop]. \tag{4.33}$$

Se confirmă astfel afirmaţia făcută mai sus că W²R este de circa n_{max}^2 ori mai rapid decât un algoritm standard care nu utilizează structura numerică specială a problemei în studiu.

Algoritmul W²R (Tabelul 4.1.) lucrează "normal" dacă şi numai dacă matricele R_0, Q_n şi S_n $(n = 1, 2, \ldots)$ sunt nesingulare. Pe de altă parte, soluţiile $\{\theta_n, \beta_n\}$ $(n = 1, 2, \ldots)$ calculate de W²R există şi sunt unic definite dacă şi numai dacă matricele $\{\Gamma_n\}$ $(n = 1, 2, \ldots)$ sunt nesingulare. Ne aşteptăm ca aceste două seturi de condiţii, garantând execuţia normală a W²R, să fie echivalente. Stabilirea acestei echivalenţe este subiectul principal al discuţiei care urmează.

Lema 4.2.1. *Fie* $\{\theta_n, Q_n\}$ *şi* $\{\beta_n, S_n\}$ *soluţiile sistemelor (4.13) şi, respectiv, (4.15)* $(n =$

Scanned with OKEN Scanner

$1, 2, \ldots$). *Să definim următoarele matrice triunghiulare:*

$$U = \begin{bmatrix} I & \theta_n & & \\ & \ddots & \ddots & \\ & & \ddots & \theta_1 \\ & & & I \end{bmatrix}, \tag{4.34}$$

$$V = \begin{bmatrix} I & 0 & \cdots & 0 \\ \beta_1 & I & & \\ & \ddots & \ddots & \\ & & \beta_n & I \end{bmatrix}. \tag{4.35}$$

Atunci matricea Γ_{n+1} *satisface următoarele ecuaţii:*

$$U\Gamma_{n+1}U^{\mathrm{T}} = \begin{bmatrix} Q_n & 0 & \cdots & 0 \\ 0 & Q_{n-1} & & \\ \vdots & & \ddots & \\ 0 & & & R_0 \end{bmatrix}, \tag{4.36}$$

$$V\Gamma_{n+1}V^{\mathrm{T}} = \begin{bmatrix} R_0 & 0 & \cdots & 0 \\ 0 & S_1 & & \\ \vdots & & \ddots & \\ 0 & & & S_n \end{bmatrix}. \tag{4.37}$$

Demonstraţie: Utilizând ecuaţia de definiţie a $\{\theta_n, Q_n\}$, (4.13), putem scrie:

$$U\Gamma_{n+1} = \begin{bmatrix} Q_n & 0 & \cdots & 0 \\ ? & Q_{n-1} & & \\ \vdots & & \ddots & \\ ? & & & R_0 \end{bmatrix}, \tag{4.38}$$

unde "?" denotă elemente a căror expresie exactă nu are importanţă pentru această demonstraţie. Utilizând (4.38) şi faptul că matricea $U\Gamma_{n+1}U^{\mathrm{T}}$ este simetrică, obţinem imediat ecuaţia (4.36). Ecuaţia (4.37) se demonstrează în mod similar.

□

Utilizând Lema 4.2.1. putem demonstra următorul rezultat.

Lema 4.2.2. *Afirmaţiile* $\{(i), (ii)$ *şi* $(iii)\}$ *şi, respectiv,* $\{(a), (b)$ *şi* $(c)\}$ *de mai jos sunt echivalente:*

$$\begin{cases} (i) & \det \Gamma_i \neq 0 & & (i = 1, 2, \ldots, n+1) \\ (ii) & \det R_0 \neq 0 & \text{şi} \quad \det Q_i \neq 0 & (i = 1, 2, \ldots, n) \\ (iii) & \det R_0 \neq 0 & \text{şi} \quad \det S_i \neq 0 & (i = 1, 2, \ldots, n) \end{cases}$$

Scanned with OKEN Scanner

$$\begin{cases} (a) & \Gamma_{n+1} > 0 \\ (b) & R_0 > 0 \quad \text{şi} \quad Q_i > 0 \quad (i = 1, 2, \ldots, n) \\ (c) & R_0 > 0 \quad \text{şi} \quad S_i > 0 \quad (i = 1, 2, \ldots, n) \end{cases}$$

Demonstraţie: În cele ce urmează vom demonstra că (i) ⇔ (ii) ⇔ (iii). Demonstraţia că (a) ⇔ (b) ⇔ (c) este perfect similară.

Implicaţia (i) ⇒ (ii) este imediată. Într-adevăr, dacă $\det \Gamma_i \neq 0$ $(i = 1, \ldots, n+1)$, atunci matricea U definită în (4.34) există. Prin urmare ecuaţia (4.36) este validă şi, deci, (ii) rezultă. Similar se arată că (i) ⇒ (iii) (utilizând ecuaţia (4.37)). Pentru a demonstra implicaţia (ii) ⇒ (i), să presupunem că (ii) este valid. Deoarece $\det R_0 \neq 0$ rezultă că θ_1 există; astfel ecuaţia (4.36) este satisfăcută pentru $n = 1$. Combinând această observaţie cu faptul că $\det Q_1 \neq 0$ (prin presupunere), urmează că $\det \Gamma_2 \neq 0$; astfel θ_2 există ş.a.m.d. Demonstraţia că (iii) ⇒ (i) este similară (singura diferenţă fiind utilizarea ecuaţiei (4.37) în locul lui (4.36)).

□

O primă implicaţie a rezultatelor demonstrate mai sus este confirmarea echivalenţei condiţiilor (i),(ii) şi (iii) care sunt necesare şi suficiente pentru execuţia normală a algoritmului W²R. Alte implicaţii interesante ale Lemelor 4.2.1. şi 4.2.2. vor fi prezentate în paragrafele următoare ale acestui capitol.

4.3. Studiul stabilităţii modelului

În acest paragraf vom arăta că, similar cazului monovariabil, stabilitatea modelului AR multivariabil determinat cu algoritmul W²R este asigurată în condiţii generale. Să notăm că modelul (4.1) este stabil dacă şi numai dacă polinomul $\det[A(z)]$ are toate zerourile în afara cercului unitar, unde

$$A(z) = I + \theta_n \begin{bmatrix} zI \\ \cdot \\ \cdot \\ \cdot \\ z^n I \end{bmatrix}. \tag{4.39}$$

Să introducem următoarea notaţie (presupunând implicit că $Q_n > 0, S_n > 0$):

$$K_{n+1} = Q_n^{-1/2} P_n S_n^{-T/2}, \tag{4.40}$$

unde $A^{1/2}$ denotă factorul Cholesky al unei matrice pozitiv definite A, iar A^{-T} este o notaţie prescurtată pentru $(A^{-1})^T = (A^T)^{-1}$. Următorul rezultat constituie extensia la cazul multivariabil a Teoremei 1.3.1.

Teorema 4.3.1. *Următoarele afirmaţii sunt echivalente:*

(I) $\Gamma_{n+1} > 0$.

(II) $R_0 > 0$ *şi* $\sigma_i < 1$ $(i = 1, \ldots, n)$, *unde* σ_i *este valoarea singulară maximă a lui* K_i.

(III) $R_0 > 0, Q_n > 0$ *şi* $\det A(z) \neq 0$ *pentru* $|z| \leq 1$.

Scanned with OKEN Scanner

Demonstraţie: Să observăm mai întâi că

$$\begin{aligned} Q_{n+1} &= Q_n - P_n S_n^{-1} P_n^{\mathrm{T}} = Q_n^{1/2}[I - Q_n^{-1/2} P_n S_n^{-\mathrm{T}/2} \cdot S_n^{-1/2} P_n^{\mathrm{T}} Q_n^{-\mathrm{T}/2}] Q_n^{\mathrm{T}/2} = \\ &= Q_n^{1/2}[I - K_{n+1} K_{n+1}^{\mathrm{T}}] Q_n^{\mathrm{T}/2} \end{aligned} \tag{4.41}$$

şi, similar

$$\begin{aligned} S_{n+1} &= S_n - P_n^{\mathrm{T}} Q_n^{-1} P_n = S_n^{1/2}[I - S_n^{-1/2} P_n^{\mathrm{T}} Q_n^{-\mathrm{T}/2} \cdot Q_n^{-1/2} P_n S_n^{-\mathrm{T}/2}] S_n^{\mathrm{T}/2} = \\ &= S_n^{1/2}[I - K_{n+1}^{\mathrm{T}} K_{n+1}] S_n^{\mathrm{T}/2}. \end{aligned} \tag{4.42}$$

Condiţia (I) este echivalentă cu (b) sau (c) din Lema 4.2.2. In mod clar, (b) sau (c) implică (II) (vezi ecuaţiile (4.41) şi (4.42) de mai sus). Reciproca este, de asemenea, adevărată: dacă $R_0 = Q_0 = S_0 > 0$ şi $\sigma_1 < 1$, atunci (4.41) şi (4.42) implică $Q_1 > 0$ şi $S_1 > 0$ care, împreună cu $\sigma_2 < 1$, implică $Q_2 > 0$ şi $S_2 > 0$ ş.a.m.d. Astfel demonstraţia că (I) şi (II) sunt echivalente este completă.

Demonstraţia echivalenţei (I) $\Leftrightarrow$ (III) este mult mai complicată. Să introducem următoarea matrice companion:

$$A_c = \left[\begin{array}{c|c} -\theta_n^{\mathrm{T}} & I \\ \hline 0 & \end{array}\right]. \tag{4.43}$$

Este binecunoscut faptul că $\det[z^n A(z^{-1})]$ este polinomul caracteristic al lui A_c[76]. Astfel, $\det[A(z)] \neq 0$ pentru $|z| \leq 1$ dacă şi numai dacă toate valorile proprii ale lui A_c sunt situate în interiorul cercului unitar. Fie λ o valoare proprie oarecare a lui A_c şi fie

$$\mu = \begin{bmatrix} \mu_1 \\ \cdot \\ \cdot \\ \cdot \\ \mu_n \end{bmatrix} \neq 0 \tag{4.44}$$

vectorul propriu asociat (unde μ_i sunt vectori $m \times 1$). Astfel:

$$A_c \mu = \lambda \mu \tag{4.45}$$

sau, într-o formă detaliată,

$$\begin{aligned} \mu_2 - A_1^{\mathrm{T}} \mu_1 &= \lambda \mu_1 \\ &\cdot \\ &\cdot \\ &\cdot \\ \mu_n - A_{n-1}^{\mathrm{T}} \mu_1 &= \lambda \mu_{n-1} \\ -A_n^{\mathrm{T}} \mu_1 &= \lambda \mu_n. \end{aligned} \tag{4.46}$$

Se poate vedea uşor că $\mu \neq 0 \Rightarrow \mu_1 \neq 0$. Într-adevăr, dacă μ_1 ar fi zero, atunci urmează imediat din (4.46) că $\mu = 0$, ceea ce ar contrazice (4.44).

Scanned with OKEN Scanner

Să observăm că setul de ecuaţii (4.46) poate fi scris în următoarea formă compactă

$$\begin{bmatrix} \mu \\ 0 \end{bmatrix} = \begin{bmatrix} I \\ \theta_n^{\mathrm{T}} \end{bmatrix} \mu_1 + \lambda \begin{bmatrix} 0 \\ \mu \end{bmatrix}. \tag{4.47}$$

Utilizând structura Toeplitz de blocuri a lui Γ_{n+1}, precum şi ecuaţiile (4.13) şi (4.47), obţinem:

$$\begin{aligned} \mu^\star \Gamma_n \mu &= \begin{bmatrix} \mu^\star & 0 \end{bmatrix} \Gamma_{n+1} \begin{bmatrix} \mu \\ 0 \end{bmatrix} = \\ &= \left\{ \mu_1^\star \begin{bmatrix} I & \theta_n \end{bmatrix} + \lambda^\star \begin{bmatrix} 0 & \mu^\star \end{bmatrix} \right\} \Gamma_{n+1} \left\{ \begin{bmatrix} I \\ \theta_n^{\mathrm{T}} \end{bmatrix} \mu_1 + \lambda \begin{bmatrix} 0 \\ \mu \end{bmatrix} \right\} = \\ &= \mu_1^\star Q_n \mu_1 + |\lambda|^2 \mu^\star \Gamma_n \mu \end{aligned} \tag{4.48}$$

(unde $\star$ denotă operaţia de transpunere şi complex conjugare). Astfel,

$$|\lambda|^2 = 1 - \frac{\mu_1^\star Q_n \mu_1}{\mu^\star \Gamma_n \mu}. \tag{4.49}$$

Cum $\Gamma_{n+1} > 0$ implică $\Gamma_n > 0$ şi $Q_n > 0$, conchidem în mod simplu din (4.49) că (I) $\Rightarrow$ (III).

Pentru a demonstra implicaţia (III) $\Rightarrow$ (I), să considerăm următoarea ecuaţie AR:

$$A(q^{-1})x(t) = w(t), \tag{4.50}$$

unde

$$Ew(t)w^{\mathrm{T}}(s) = Q_n \delta_{t,s} \tag{4.51}$$

(Notaţi că Q_n este o matrice de covarianţă validă, adică $Q_n > 0$, prin presupunere). Deoarece ecuaţia AR, (4.50), este stabilă prin presupunere, $x(t)$ poate fi scris ca o medie alunecătoare infinită în $\{w(t)\}$:

$$x(t) = w(t) + C_1 w(t-1) + C_2 w(t-2) + \ldots \tag{4.52}$$

ceea ce implică următoarele egalităţi

$$\begin{aligned} Ew(t)x^{\mathrm{T}}(t) &= Q_n \\ Ew(t)x^{\mathrm{T}}(t-k) &= 0 \quad \text{pentru } k \geq 1. \end{aligned} \tag{4.53}$$

În acest moment al demonstraţiei este necesar să introducem secvenţa de covarianţe asociată cu $x(t)$:

$$R_k^x = Ex(t)x^{\mathrm{T}}(t-k) \quad (k = 0, \pm 1, \pm 2, \ldots).$$

Dacă multiplicăm (4.50) la dreapta cu $x^{\mathrm{T}}(t-k)$ $(k = 0, 1, 2, \ldots)$, aplicăm operatorul de mediere şi utilizăm ecuaţiile (4.53), vom obţine următoarele relaţii:

$$R_0^x + A_1 R_{-1}^x + \ldots + A_n R_{-n}^x = Q_n \tag{4.54}$$

$$\theta_n \Gamma_n^x = -\gamma_n^x \tag{4.55}$$

şi

$$R_{n+k}^x = -A_1 R_{n+k-1}^x - \ldots - A_n R_k^x \quad (k \geq 1), \tag{4.56}$$

unde Γ_n^x şi γ_n^x sunt definite similar cu Γ_n şi, respectiv, γ_n. Soluţia $\{R_0^x, \ldots, R_n^x\}$ a ecuaţiilor (4.54), (4.55) este unică. Într-adevăr, dacă ar exista două soluţii,atunci ar conduce, prin (4.56), la două secvenţe diferite de covarianţe asociate aceleaşi ecuaţii AR, ceea ce este imposibil.

Să ne reamintim acum că $\{A_i\}$ în (4.54),(4.55) satisfac (4.13). Este atunci uşor de văzut că soluţia unică a ecuaţiilor (4.54), (4.55) este

$$R_k^x = R_k, \quad k = 0, \ldots, n. \tag{4.57}$$

Cum secvenţa $\{R_k^x\}_{k=0,1,\ldots}$ este pozitiv definită prin construcţie, rezultă că (III) $\Rightarrow$ (I) şi demonstraţia este terminată.

Comparând Teorema 4.3.1. cu Teorema 1.3.1., observăm că există un paralelism aproape perfect între ele; singura excepţie aparentă de la acest paralelism este apariţia unei condiţii adiţionale $Q_n > 0$ care apare în cazul multivariabil, dar nu se regăseşte în cazul monovariabil. Totuşi, explicaţia acestei diferenţe este simplă: singura alternativă pentru $Q_n \neq 0$ în cazul scalar este $Q_n > 0$, pe când în cazul multivariabil apare în plus posibilitatea ca $Q_n \geq 0$ (pozitiv semidefinită şi singulară), care nu există în cazul monovariabil şi care trebuie exclusă prin presupuneri adecvate.

Teorema 4.3.1. are un număr de implicaţii importante. Parte dintre acestea sunt discutate în paragraful următor. Aici ne vom referi numai la implicaţiile privind stabilitatea modelului AR determinat cu algoritmul W^{2}R.

Conform cu Teorema 4.3.1., condiţia $\Gamma_{n+1} > 0$ este necesară şi suficientă pentru stabilitatea ecuaţiei AR. Această condiţie asupra lui Γ_{n+1} poate fi uşor satisfăcută în aplicaţii utilizând formula (4.10) pentru calculul covarianţelor de eşantion. (demonstraţia acestui fapt este perfect similară celei prezentate în cazul scalar, vezi ecuaţia (1.45)). Putem, deci, conchide că *modelul AR determinat cu algoritmul W^{2}R prin prelucrarea covarianţelor de eşantion (4.10) este stabil.*

În privinţa "gradului de stabilitate" al modelului AR furnizat de W^{2}R, este posibilă demonstrarea unui rezultat similar rezultatului relevant din cazul scalar (Teorema 1.3.2.).

În fine, să notăm că în cursul demonstraţiei implicaţiei (III) $\Rightarrow$ (I) am obţinut următorul rezultat cu interes de sine stătător: *Dacă $\Gamma_{n+1} > 0$, atunci ecuaţia AR (4.50), unde $\{A_i\}$ şi Q_n sunt obţinute cu relaţia (4.13), sintetizează în mod exact secvenţa dată de covarianţe $\{R_0, R_1, \ldots, R_n\}$.* Acest rezultat are un număr de aplicaţii. De exemplu, el poate fi utilizat pentru a obţine o extensie infinită pozitiv definită a unei secvenţe finite de covarianţe. O altă consecinţă interesantă priveşte posibilitatea de a parametriza orice secvenţă pozitiv definită de covarianţe prin parametrii modelului AR asociat. Această posibilitate ar putea oferi o soluţie unei probleme importante: estimarea secvenţelor de covarianţă.

4.4. Utilizări ale algoritmului Whittle–Wiggins–Robinson

Algoritmul W^{2}R are, după cum am văzut, proprietăţi similare algoritmului LD. Ne putem deci aştepta ca şi aplicaţiile celor doi algoritmi să fie similare (pentru ALD vezi paragraful 1.4). O primă utilizare a AW^{2}R, descrisă în detaliu în paragrafele anterioare, este *determinarea rapidă a parametrilor modelelor AR multivariabile* prin prelucrarea covarianţelor de eşantion ale datelor. Alte aplicaţii posibile ale AW^{2}R sunt descrise succint în cele ce urmează.

Scanned with OKEN Scanner

4.4.1. Factorizarea matricelor inverse de covarianţă

Urmează imediat din Lema 4.2.1. că matricea inversă de covarianţă Γ_{n+1}^{-1} admite următoarele două factorizări: o factorizare inferior-superior triunghiulară,

$$\Gamma_{n+1}^{-1} = U^{\mathrm{T}} \begin{bmatrix} Q_n^{-1} & 0 & \cdot & & 0 \\ 0 & Q_{n-1}^{-1} & & & \\ \cdot & & \cdot & & \\ \cdot & & & Q_1^{-1} & 0 \\ 0 & \cdot & & 0 & R_0^{-1} \end{bmatrix} U \tag{4.58}$$

şi una superior-inferior triunghiulară,

$$\Gamma_{n+1}^{-1} = V^{\mathrm{T}} \begin{bmatrix} R_0^{-1} & 0 & \cdot & & 0 \\ 0 & S_1^{-1} & & & \\ \cdot & & \cdot & & \\ \cdot & & & S_{n-1}^{-1} & 0 \\ 0 & \cdot & & 0 & S_n^{-1} \end{bmatrix} V. \tag{4.59}$$

Factorizările (4.58) şi (4.59) sunt generalizări la cazul multivariabil ale factorizărilor (1.71) şi, respectiv, (2.18) valide în cazul monovariabil. Toate cantităţile care apar în (4.58), (4.59) se obţin în cadrul AW²R. Astfel, determinarea factorizărilor (4.58) şi (4.59) nu cere nici un calcul în plus faţă de cele efectuate în mod normal de AW²R. Pentru determinarea factorilor Cholesky ai lui Γ_{n+1}^{-1}, unele calcule adiţionale sunt, totuşi, necesare (vezi mai jos).

Fie $Q_i^{-1/2}$ factorul Cholesky superior triunghiular al lui Q_i^{-1} $(i = 0, \ldots, n;\ Q_0 = R_0)$. Atunci

$$\begin{bmatrix} Q_n^{-1/2} & 0 & \cdot & & 0 \\ 0 & Q_{n-1}^{-1/2} & & & \\ \cdot & & \cdot & & \\ \cdot & & & Q_1^{-1/2} & 0 \\ 0 & \cdot & & 0 & Q_0^{-1/2} \end{bmatrix} U \tag{4.60}$$

este factorul Cholesky superior-triunghiular al lui Γ_{n+1}^{-1} (vezi (4.58)). Similar, urmează din (4.59) că factorul Cholesky inferior-triunghiular al lui Γ_{n+1}^{-1} este

$$\begin{bmatrix} S_0^{-1/2} & 0 & \cdot & & 0 \\ 0 & S_1^{-1/2} & & & \\ \cdot & & \cdot & & \\ \cdot & & & S_{n-1}^{-1/2} & 0 \\ 0 & \cdot & & 0 & S_n^{-1/2} \end{bmatrix} V, \tag{4.61}$$

unde $S_i^{-1/2}$ este factorul Cholesky inferior-triunghiular al lui S_i^{-1} $(i = 0, \ldots, n;\ S_0 = R_0)$. Observăm că operaţia adiţională necesară atât în (4.60), cât şi în (4.61), constă în determinarea

factorilor Cholesky ai $(n+1)$ matrice pozitiv definite de dimensiune $m \times m$ (o operaţie mult mai puţin costisitoare decât evaluarea factorului Cholesky al unei matrice de dimensiune $[m \cdot (n+1)] \times [m \cdot (n+1)]$).

4.4.2. Testarea pozitivităţii matricelor Toeplitz de blocuri

Conform Lemei 4.2.2., matricea Toeplitz de blocuri Γ_{n+1} este pozitiv definită dacă şi numai dacă

$$Q_i > 0 \quad (i = 0, 1, \ldots, n; \quad Q_0 = R_0). \tag{4.62}$$

Testarea pozitivităţii unei matrice $(n \cdot m) \times (n \cdot m)$ poate, astfel, să fie redusă la testarea pozitivităţii a n matrice $m \times m$; procedând în acest mod, timpul de calcul este redus considerabil. Condiţia $Q_i > 0$ poate fi testată calculând valorile proprii ale lui Q_i şi verificând dacă toate sunt strict pozitive, sau nu.

4.4.3. Testarea stabilităţii

Să introducem următoarele două polinoame matriceale:

$$A_n(z) = I + \theta_n \begin{bmatrix} zI \\ . \\ . \\ . \\ z^n I \end{bmatrix}, \tag{4.63}$$

$$B_n(z) = \beta_n \begin{bmatrix} I \\ . \\ . \\ . \\ Iz^{n-1} \end{bmatrix} + Iz^n. \tag{4.64}$$

Ecuaţiile recurente de bază (4.22),(4.23) ale AW^{2}R pot fi scrise sub următoarea formă:

$$\begin{bmatrix} I & \theta_{n+1} \\ \beta_{n+1} & I \end{bmatrix} = \begin{bmatrix} I & \begin{bmatrix} \theta_n & 0 \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} \beta_n & I \end{bmatrix} \\ \tilde{K}_{n+1}^{\mathrm{T}} \begin{bmatrix} I & \theta_n \end{bmatrix} + \begin{bmatrix} 0 & \beta_n \end{bmatrix} & I \end{bmatrix}. \tag{4.65}$$

Multiplicând (4.65) la dreapta cu $\begin{bmatrix} I & Iz & \ldots & Iz^{n+1} \end{bmatrix}^{\mathrm{T}}$, obţinem următoarea ecuaţie recursivă polinomială:

$$\begin{bmatrix} A_{n+1}(z) \\ B_{n+1}(z) \end{bmatrix} = \begin{bmatrix} I & \overline{K}_{n+1} \\ \tilde{K}_{n+1}^{\mathrm{T}} & I \end{bmatrix} \begin{bmatrix} A_n(z) \\ zB_n(z) \end{bmatrix}. \tag{4.66}$$

Ecuaţia (4.66) este corespondentul multivariabil al ecuaţiei (1.77) validă în cazul scalar. Similar cazului scalar, ecuaţia (4.66) ar trebui "inversată" pentru a putea utiliza Teorema 4.3.1. ca bază în conceperea unui test de stabilitate.

Utilizând Lema 3.2.2. putem scrie:

$$\begin{bmatrix} I & \overline{K}_{n+1} \\ \tilde{K}_{n+1}^{\mathrm{T}} & I \end{bmatrix}^{-1} = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -\overline{K}_{n+1} \\ I \end{bmatrix} (I - \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1})^{-1} \begin{bmatrix} -\tilde{K}_{n+1}^{\mathrm{T}} & I \end{bmatrix} \tag{4.67}$$

$$= \begin{bmatrix} I + \overline{K}_{n+1}(I - \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1})^{-1} \tilde{K}_{n+1}^{\mathrm{T}} & -\overline{K}_{n+1}(I - \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1})^{-1} \\ -(I - \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1})^{-1} \tilde{K}_{n+1}^{\mathrm{T}} & (I - \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1})^{-1} \end{bmatrix}. \tag{4.68}$$

Scanned with OKEN Scanner

Să observăm acum că:

$$\begin{aligned} \tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1} &= P_n^{\mathrm{T}} Q_n^{-1} P_n S_n^{-1} = \\ &= S_n^{1/2} \left(S_n^{-1/2} P_n^{\mathrm{T}} Q_n^{-\mathrm{T}/2} \right) \left(Q_n^{-1/2} P_n S_n^{-\mathrm{T}/2} \right) S_n^{-1/2} = \\ &= S_n^{1/2} \left(K_{n+1}^{\mathrm{T}} K_{n+1} \right) S_n^{-1/2}. \end{aligned} \tag{4.69}$$

Astfel matricele $\tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1}$ şi $K_{n+1}^{\mathrm{T}} K_{n+1}$ sunt legate printr-o transformare de similaritate. Rezultă că

$$\text{val. propr. } \{\tilde{K}_{n+1}^{\mathrm{T}} \overline{K}_{n+1}\} = \text{val. propr. } \{K_{n+1}^{\mathrm{T}} K_{n+1}\} = \{\text{val. sing. } (K_{n+1})\}^2. \tag{4.70}$$

Prin urmare, condiţia $\sigma_i < 1 \quad (i = 1, \ldots, n)$ care apare în Teorema 4.3.1. este echivalentă cu:

$$\lambda_i < 1 \quad (i = 1, \ldots, n), \tag{4.71}$$

unde $\lambda_i =$ val. propr. max. $(\tilde{K}_i^{\mathrm{T}} \overline{K}_i)$.

Suntem acum în măsură să formulăm un test de stabilitate bazat pe Teorema 4.3.1.:

a) Polinomul $A_n(z)$ (a cărui stabilitate trebuie testată) este dat. Determinaţi polinomul asociat $B_n(z)$.

b) Pornind de la $\{A_n(z), B_n(z)\}$ utilizaţi algoritmul W²R *inversat* pentru deducerea $\{A_i(z), B_i(z)\}_{i=n-1,n-2,\ldots}$ (prin reducerea succesivă a gradului polinoamelor iniţiale $\{A_n(z), B_n(z)\}$). Algoritmul W²R inversat constă din ecuaţia (4.66) inversată (utilizând (4.67)) şi din următoarele ecuaţii (vezi (4.22),(4.23)):

$$\begin{aligned} \overline{K}_n &= \text{ultimul bloc } m \times m \text{ al lui } \theta_n, \\ \tilde{K}_{n+1}^{\mathrm{T}} &= \text{primul bloc } m \times m \text{ al lui } \beta_n. \end{aligned} \tag{4.72}$$

c) Calculaţi valorile proprii maxime $\{\lambda_i\}$ ale matricelor $\{\tilde{K}_i^{\mathrm{T}} \overline{K}_i\}$. Dacă $\{\lambda_i\}$ satisfac (4.71), atunci polinomul $A_n(z)$ este stabil. Dacă nu, polinomul este instabil.

Singurul pas care nu a fost explicitat mai sus, este cel referitor la calculul lui $B_n(z)$ din $A_n(z)$. În cazul scalar acest pas era trivial: simpla reversare a vectorului coeficienţilor lui $A_n(z)$ producea vectorul coeficienţilor lui $B_n(z)$ (vezi (1.78)). În cazul multivariabil problema determinării lui $B_n(z)$ din $A_n(z)$ este mult mai complicată. Algoritmi care efectuează acest calcul sunt descrişi în [33],[53],[94].

În încheierea acestui paragraf să remarcăm că testul de stabilitate descris mai sus este versiunea multivariabilă a procedurii Schur–Cohn prezentate în Capitolul 1 [94],[96].

Scanned with OKEN Scanner

Capitolul 5

Algoritmul Levinson pentru Modele Multivariabile

În următoarele două capitole (5 şi 6) vom prezenta algoritmi rapizi pentru estimarea parametrilor celor mai frecvent utilizate tipuri de *modele de sistem*. În contrast cu *modelele de semnal* considerate până acum, modelele de sistem descriu influenţa asupra ieşirii nu numai a unor variabile asupra cărora nu avem un control direct (variabile desemnate prin denumirea generică de "zgomot"), ci şi a unei *intrări de control*. Modelele de sistem sunt instrumente indispensabile *în proiectarea sistemelor de conducere* în economie şi tehnică, în studii de *prognoză şi simulare* etc. În continuare vom considera direct modele de sisteme multivariabile; particularizarea rezultatelor ce vor fi prezentate la cazul monovariabil este lăsată ca un exerciţiu simplu pentru cititor.

5.1. Modele de tip sumă de convoluţie

Fie $\{y(t)\}_{t=1,2,\ldots}$ ieşirea m_y–dimensională a unui sistem liniar multivariabil, iar $\{u(t)\}$ intrarea m_u–dimensională a acestuia. Sistemul poate fi descris prin următoarea ecuaţie de tip sumă de convoluţie:

$$y(t) = \sum_{k=1}^{\infty} G_k u(t-k) + \varepsilon(t), \tag{5.1}$$

unde $\{G_k\}$ este secvenţa de ponderare a sistemului, iar $\varepsilon(t)$ este un vector m_y–dimensional de zgomot. Problema de modelare corespunzătoare ecuaţiei (5.1) este determinarea secvenţei de ponderare din măsurători intrare–ieşire (să notăm că $\{G_k\}$ mai sunt numiţi şi coeficienţi/parametri Markov ai sistemului).

În aplicaţii modelul (5.1) este aproape întotdeauna utilizat sub următoarea formă *trunchiată*:

$$y(t) = \sum_{k=1}^{n} G_k u(t-k) + \varepsilon(t), \tag{5.2}$$

unde n este astfel ales încât $G_k = 0$ (sau $G_k \approx 0$) pentru $k > n$. Majoritatea proceselor practice au "memorie" finită; cu alte cuvinte $u(t-k)$, $k \gg 1$, nu mai influenţează ieşirea $y(t)$ la momentul t, sau o influenţează extrem de puţin. În astfel de cazuri, ecuaţia (5.2), cu valori rezonabile ale lui n (de ordinul zecilor), constituie un model corespunzător al dinamicii sistemului liniar în studiu. Să observăm că (5.2) presupune implicit că nu există transmisie instantanee de la intrare la ieşire (adică $G_0 = 0$). Am făcut această presupunere pentru că ea corespunde cazului celui mai

Scanned with OKEN Scanner

frecvent întâlnit în practică. Dacă, totuşi, sistemul analizat are transmisie instantanee, sau un timp mort diferit de o perioadă de eşantionare (ca în (5.2)), modelul (5.2) poate fi uşor adaptat prin simpla translaţie a semnalului de intrare. Pentru a fi mai exacţi, să presupunem că timpul mort (sau "întârzierea pură") a sistemului este egal cu τ (≥ 0). Atunci, renotând/translatând semnalul de intrare în următorul mod

$$u(t-\tau) \rightarrow u(t-1), \tag{5.3}$$

putem utiliza ecuaţia (5.2) pentru a modela sistemul.

Să considerăm în continuare problema determinării coeficienţilor matriceali $\{G_k\}_{k=1}^n$ ai secvenţei de ponderare şi, de asemenea, a matricei de covarianţă a zgomotului:

$$Q = E\varepsilon(t)\varepsilon^{\mathrm{T}}(t). \tag{5.4}$$

Vom utiliza în acest scop tot metoda CMMP. Cu alte cuvinte, vom determina G_k prin minimizarea criteriului matriceal (5.4) în raport cu aceşti parametri, iar pe Q îl vom lua egal cu valoarea lui (5.4) în punctul de minim. Deducerea estimaţiei CMMP a parametrilor $\{G_k\}$ şi Q este similară celei prezentate în paragraful 4.1. Să notăm totuşi, că în cazul de faţă problema de estimare este puţin mai complexă deoarece trebuie să operăm cu două semnale $\{y(t)\}$ şi $\{u(t)\}$ şi nu cu un singur semnal $\{y(t)\}$ ca în cazul ecuaţiilor AR multivariabile considerate în Capitolul 4.

Pentru a deduce estimaţiile CMMP ale $\{G_k\}$ şi Q este necesar să introducem următoarele notaţii:

$$R_k = Eu(t)u^{\mathrm{T}}(t-k) = R_{-k}^{\mathrm{T}}, \tag{5.5}$$

$$S_k = Ey(t)u^{\mathrm{T}}(t-k), \tag{5.6}$$

$$\Lambda_k = Ey(t)y^{\mathrm{T}}(t-k), \tag{5.7}$$

$$\gamma = Ey(t)\left[\, u^{\mathrm{T}}(t-1) \;\; \ldots \;\; u^{\mathrm{T}}(t-n) \,\right] = \left[\, S_1 \;\; \ldots \;\; S_n \,\right], \tag{5.8}$$

$$\begin{aligned} \Gamma &= E\begin{bmatrix} u(t-1) \\ \cdot \\ \cdot \\ \cdot \\ u(t-n) \end{bmatrix}\left[\, u^{\mathrm{T}}(t-1) \;\; \ldots \;\; u^{\mathrm{T}}(t-n) \,\right] = \\ &= \begin{bmatrix} R_0 & R_1 & \cdot & \cdot & \cdot & R_{n-1} \\ R_{-1} & R_0 & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ R_{1-n} & \cdot & \cdot & \cdot & \cdot & R_0 \end{bmatrix}, \end{aligned} \tag{5.9}$$

$$\theta = \left[\, G_1 \;\; \ldots \;\; G_n \,\right]. \tag{5.10}$$

Utilizând aceste notaţii, putem scrie Q sub următoarea formă:

$$Q = E\left\{ y(t) - \theta \begin{bmatrix} u(t-1) \\ \cdot \\ \cdot \\ u(t-n) \end{bmatrix} \right\} \left\{ y^{\mathrm{T}}(t) - \left[\, u^{\mathrm{T}}(t-1) \;\; \ldots \;\; u^{\mathrm{T}}(t-n) \,\right]\theta^{\mathrm{T}} \right\} =$$

Scanned with OKEN Scanner

$$= \Lambda_0 - \gamma\theta^T - \theta\gamma^T + \theta\Gamma\theta^T =$$
$$= (\theta - \gamma\Gamma^{-1})\Gamma(\theta - \gamma\Gamma^{-1})^T + \Lambda_0 - \gamma\Gamma^{-1}\gamma^T. \quad (5.11)$$

Deoarece matricea Γ este pozitiv definită (vezi paragraful 4.1), rezultă imediat că Q este minimizat pentru

$$\theta = \gamma\Gamma^{-1}, \quad (5.12)$$

valoarea minimă corespunzătoare fiind

$$Q = \Lambda_0 - \gamma\Gamma^{-1}\gamma^T. \quad (5.13)$$

Ecuaţia (5.12) pentru determinarea lui θ este celebra *ecuaţie Wiener-Hopf*. Ea are aceeaşi formă cu ecuaţia (4.8) corespunzătoare modelelor AR. Totuşi, (5.12) este mai complexă decât (4.8) deoarece,spre deosebire de (4.8) care nu conţine decât funcţia de autocovarianţă a semnalului $y(t)$, (5.12) include atât funcţia de autocovarianţă a intrării, cât şi funcţia de inter-covarianţă intrare–ieşire. Să notăm, aşa cum am făcut-o de fiecare dată, că în aplicaţii funcţiile teoretice de covarianţă care apar în (5.12) sunt înlocuite cu funcţii de covarianţă de eşantion:

$$R_k = \frac{1}{N}\sum_{t=k+1}^{N} u(t)u^T(t-k) = R_{-k}^T, \quad (5.14)$$

$$S_k = \frac{1}{N}\sum_{t=k+1}^{N} y(t)u^T(t-k), \quad (5.15)$$

$$\Lambda_0 = \frac{1}{N}\sum_{t=k+1}^{N} y(t)y^T(t) \quad (5.16)$$

(pentru care păstrăm notaţiile asociate cu funcţiile teoretice de covarianţă, pentru a nu complica inutil prezentarea).

În aplicaţii este deseori necesară rezolvarea sistemului de ecuaţii liniare (5.12) pentru $n = 1, 2, \ldots, n_{max}$. Un algoritm standard utilizat în acest scop ar necesita $\mathcal{O}(m_u^3 n_{max}^4)$ operaţii aritmetice. Acest efort considerabil de calcul poate fi drastic redus, de circa n_{max}^2 ori, dacă exploatăm structura specială a sistemului de ecuaţii (5.12). Deducerea unui algoritm rapid pentru determinarea lui $\{G_k\}$ şi Q ($n = 1, 2, \ldots$) este subiectul paragrafului următor. Să notăm că algoritmul ce va fi prezentat în cele ce urmează a fost introdus în literatura de specialitate (pentru cazul modelelor monovariabile de tip sumă de convoluţie) de către Levinson[39] (extensia la cazul multivariabil a apărut în [1]).

5.2. Algoritmul Levinson

Vom ataşa un indice n cantităţilor ce apar în (5.12), (5.13) etc. pentru a sublinia faptul că ele corespund modelului de ordin n:

$$\theta_n = \gamma_n\Gamma_n^{-1}, \quad (5.17)$$
$$Q_n = \Lambda_0 - \gamma_n\Gamma_n^{-1}\gamma_n^T. \quad (5.18)$$

Similar cazului AR multivariabil (tratat în Capitolul 4), Γ_n este o *matrice simetrică Toeplitz de blocuri*. Totuşi, spre deosebire de cazul AR multivariabil, în cazul de faţă între blocurile lui γ_n

Scanned with OKEN Scanner

şi cele ale lui Γ_n nu mai există nici o legătura directă. Acesta este motivul principal pentru care AL corespunzător modelelor sumă de covoluţie este mai complex, din punct de vedere numeric, decât AL corespunzător modelelor AR multivariabile.

Rezolvarea eficientă (recursivă-în-n) a sistemului de ecuaţii liniare (5.17) cere introducerea unui număr de variabile auxiliare care sunt definite în continuare:

$$\begin{aligned}
\psi_n &= \begin{bmatrix} R_n \\ \cdot \\ \cdot \\ \cdot \\ R_1 \end{bmatrix}, \\
\phi_n &= \begin{bmatrix} R_{-1} \\ \cdot \\ \cdot \\ \cdot \\ R_{-n} \end{bmatrix}, \\
\Delta_n &= \psi_n^{\mathrm{T}}\Gamma_n^{-1}, \\
\beta_n &= \phi_n^{\mathrm{T}}\Gamma_n^{-1}, \\
\Omega_n &= S_{n+1} - \gamma_n\Gamma_n^{-1}\psi_n = S_{n+1} - \theta_n\psi_n, \\
P_n &= R_{-(n+1)} - \psi_n^{\mathrm{T}}\Gamma_n^{-1}\phi_n = R_{-(n+1)} - \Delta_n\phi_n, \\
\pi_n &= R_0 - \phi^{\mathrm{T}}\Gamma_n^{-1}\phi_n, \\
\Sigma_n &= R_0 - \psi_n^{\mathrm{T}}\Gamma_n^{-1}\psi_n.
\end{aligned} \tag{5.19}$$

Partea principală a AL constă dintr-un set de ecuaţii care determină soluţia de ordin $(n+1)$ din cea de ordin n. Acest set de ecuaţii este dedus în următoarea teoremă.

Teorema 5.2.1. *Soluţiile de ordin n şi $(n+1)$ ale sistemului liniar de ecuaţii (5.17), (5.18) sunt legate prin următoarele relaţii:*

$$\theta_{n+1} = \begin{bmatrix} \theta_n & 0 \end{bmatrix} + K_{n+1}\begin{bmatrix} -\Delta_n & I \end{bmatrix}, \tag{5.20}$$

$$\Delta_{n+1} = \begin{bmatrix} 0 & \Delta_n \end{bmatrix} + \overline{K}_{n+1}\begin{bmatrix} I & -\beta_n \end{bmatrix}, \tag{5.21}$$

$$\beta_{n+1} = \begin{bmatrix} \beta_n & 0 \end{bmatrix} + \tilde{K}_{n+1}\begin{bmatrix} -\Delta_n & I \end{bmatrix}, \tag{5.22}$$

$$\pi_{n+1} = \pi_n - \tilde{K}_{n+1}P_n, \tag{5.23}$$

$$\Sigma_{n+1} = \Sigma_n - \overline{K}_{n+1}P_n^{\mathrm{T}}, \tag{5.24}$$

$$Q_{n+1} = Q_n - K_{n+1}\Omega_n^{\mathrm{T}}, \tag{5.25}$$

unde

$$K_{n+1} = \Omega_n\Sigma_n^{-1}, \tag{5.26}$$

$$\overline{K}_{n+1} = P_n\pi_n^{-1}, \tag{5.27}$$

$$\tilde{K}_{n+1} = P_n^{\mathrm{T}}\Sigma_n^{-1}. \tag{5.28}$$

Demonstraţie: Vom începe prin a nota următoarele partiţionări matriceale:

$$\Gamma_{n+1} = \begin{bmatrix} \Gamma_n & \psi_n \\ \psi_n^{\mathrm{T}} & R_0 \end{bmatrix} = \begin{bmatrix} R_0 & \phi_n^{\mathrm{T}} \\ \phi_n & \Gamma_n \end{bmatrix}, \tag{5.29}$$

Scanned with OKEN Scanner

$$\psi_{n+1}^{\mathrm{T}} = \left[\begin{array}{cc} R_{-(n+1)} & \psi_n^{\mathrm{T}} \end{array} \right], \tag{5.30}$$

$$\phi_{n+1}^{\mathrm{T}} = \left[\begin{array}{cc} \phi_n^{\mathrm{T}} & R_{n+1} \end{array} \right], \tag{5.31}$$

$$\gamma_{n+1} = \left[\begin{array}{cc} \gamma_n & S_{n+1} \end{array} \right]. \tag{5.32}$$

Utilizând Lema 3.2.2. şi (5.29)–(5.32), obţinem următoarea ecuaţie recursivă pentru θ_{n+1}:

$$\begin{aligned} \theta_{n+1} &= \gamma_{n+1}\Gamma_{n+1}^{-1} = \left[\begin{array}{cc} \gamma_n & S_{n+1} \end{array} \right] \left\{ \left[\begin{array}{c} I \\ 0 \end{array} \right] \Gamma_n^{-1} \left[\begin{array}{cc} I & 0 \end{array} \right] + \right. \\ &+ \left. \left[\begin{array}{c} -\Gamma_n^{-1}\psi_n \\ I \end{array} \right] (R_0 - \psi_n^{\mathrm{T}}\Gamma_n^{-1}\psi_n)^{-1} \left[\begin{array}{cc} -\psi_n^{\mathrm{T}}\Gamma_n^{-1} & I \end{array} \right] \right\} = \\ &= \left[\begin{array}{cc} \theta_n & 0 \end{array} \right] + \Omega_n\Sigma_n^{-1} \left[\begin{array}{cc} -\Delta_n & I \end{array} \right] = \left[\begin{array}{cc} \theta_n & 0 \end{array} \right] + K_{n+1} \left[\begin{array}{cc} -\Delta_n & I \end{array} \right] \end{aligned} \tag{5.33}$$

care coincide cu (5.20). Procedând similar, obţinem următoarele ecuaţii recurente pentru $\{\Delta_n\}, \{\beta_n\}, \{\pi_n\}, \{\Sigma_n\}$ şi $\{Q_n\}$:

$$\begin{aligned} \Delta_{n+1} &= \psi_{n+1}^{\mathrm{T}}\Gamma_{n+1}^{-1} = \left[\begin{array}{cc} R_{-(n+1)} & \psi_n^{\mathrm{T}} \end{array} \right] \left\{ \left[\begin{array}{c} 0 \\ I \end{array} \right] \Gamma_n^{-1} \left[\begin{array}{cc} 0 & I \end{array} \right] + \right. \\ &+ \left. \left[\begin{array}{c} I \\ -\Gamma_n^{-1}\phi_n \end{array} \right] (R_0 - \phi_n^{\mathrm{T}}\Gamma_n^{-1}\phi_n)^{-1} \left[\begin{array}{cc} I & -\phi_n^{\mathrm{T}}\Gamma_n^{-1} \end{array} \right] \right\} = \\ &= \left[\begin{array}{cc} 0 & \Delta_n \end{array} \right] + P_n\pi_n^{-1} \left[\begin{array}{cc} I & -\beta_n \end{array} \right] = \left[\begin{array}{cc} 0 & \Delta_n \end{array} \right] + \overline{K}_{n+1} \left[\begin{array}{cc} I & -\beta_n \end{array} \right], \end{aligned} \tag{5.34}$$

$$\begin{aligned} \beta_{n+1} &= \phi_{n+1}^{\mathrm{T}}\Gamma_{n+1}^{-1} = \left[\begin{array}{cc} \phi_n^{\mathrm{T}} & R_{n+1} \end{array} \right] \left\{ \left[\begin{array}{c} I \\ 0 \end{array} \right] \Gamma_n^{-1} \left[\begin{array}{cc} I & 0 \end{array} \right] + \right. \\ &+ \left. \left[\begin{array}{c} -\Gamma_n^{-1}\psi_n \\ I \end{array} \right] \Sigma_n^{-1} \left[\begin{array}{cc} -\psi_n^{\mathrm{T}}\Gamma_n^{-1} & I \end{array} \right] \right\} = \\ &= \left[\begin{array}{cc} \beta_n & 0 \end{array} \right] + P_n\Sigma_n^{-1} \left[\begin{array}{cc} -\Delta_n & I \end{array} \right] = \left[\begin{array}{cc} \beta_n & 0 \end{array} \right] + \tilde{K}_{n+1} \left[\begin{array}{cc} -\Delta_n & I \end{array} \right], \end{aligned} \tag{5.35}$$

$$\begin{aligned} \pi_{n+1} &= R_0 - \phi_{n+1}^{\mathrm{T}}\Gamma_{n+1}^{-1}\phi_{n+1} = R_0 - \left[\begin{array}{cc} \phi_n^{\mathrm{T}} & R_{n+1} \end{array} \right] \left\{ \left[\begin{array}{c} I \\ 0 \end{array} \right] \Gamma_n^{-1} \left[\begin{array}{cc} I & 0 \end{array} \right] + \right. \\ &+ \left. \left[\begin{array}{c} -\Gamma_n^{-1}\psi_n \\ I \end{array} \right] \Sigma_n^{-1} \left[\begin{array}{cc} -\psi_n^{\mathrm{T}}\Gamma_n^{-1} & I \end{array} \right] \right\} \left[\begin{array}{c} \phi_n \\ R_{-(n+1)} \end{array} \right] = \\ &= \pi_n - P_n^{\mathrm{T}}\Sigma_n^{-1}P_n = \pi_n - \tilde{K}_{n+1}P_n, \end{aligned} \tag{5.36}$$

$$\begin{aligned} \Sigma_{n+1} &= R_0 - \psi_{n+1}^{\mathrm{T}}\Gamma_{n+1}^{-1}\psi_{n+1} = R_0 - \left[\begin{array}{cc} R_{-(n+1)} & \psi_n^{\mathrm{T}} \end{array} \right] \left\{ \left[\begin{array}{c} 0 \\ I \end{array} \right] \Gamma_n^{-1} \left[\begin{array}{cc} 0 & I \end{array} \right] + \right. \\ &+ \left. \left[\begin{array}{c} I \\ -\Gamma_n^{-1}\phi_n \end{array} \right] \pi_n^{-1} \left[\begin{array}{cc} I & -\phi_n^{\mathrm{T}}\Gamma_n^{-1} \end{array} \right] \right\} \left[\begin{array}{c} R_{n+1} \\ \psi_n \end{array} \right] = \\ &= \Sigma_n - P_n\pi_n^{-1}P_n^{\mathrm{T}} = \Sigma_n - \overline{K}_{n+1}P_n^{\mathrm{T}}, \end{aligned} \tag{5.37}$$

Scanned with OKEN Scanner

$$
\begin{aligned}
Q_{n+1} &= \Lambda_0 - \gamma_{n+1}\Gamma_{n+1}^{-1}\gamma_{n+1}^{\mathrm{T}} = \Lambda_0 - \begin{bmatrix} \gamma_n & S_{n+1} \end{bmatrix} \left\{ \begin{bmatrix} I \\ 0 \end{bmatrix} \Gamma_n^{-1} \begin{bmatrix} I & 0 \end{bmatrix} + \right. \\
&+ \left. \begin{bmatrix} -\Gamma_n^{-1}\psi_n \\ I \end{bmatrix} \Sigma_n^{-1} \begin{bmatrix} -\psi_n^{\mathrm{T}}\Gamma_n^{-1} & I \end{bmatrix} \right\} \begin{bmatrix} \gamma_n^t \\ S_{-(n+1)}^{\mathrm{T}} \end{bmatrix} = \\
&= Q_n - \Omega_n\Sigma_n^{-1}\Omega_n^{\mathrm{T}} = Q_n - K_{n+1}\Omega_n^{\mathrm{T}}, \qquad (5.38)
\end{aligned}
$$

care coincid, respectiv, cu (5.21)–(5.25). Demonstraţia este, deci, completă.

□

Ecuaţiile recurente (5.20)–(5.25) ale AL trebuie iniţializate cu $\theta_1, \Delta_1, \beta_1, \pi_1, \Sigma_1$ şi Q_1. Aceste valori iniţiale se obţin uşor din ecuaţiile de definiţie corespunzătoare:

$$
\begin{aligned}
\theta_1 &= S_1 R_0^{-1}, & \Delta_1 &= R_1^{\mathrm{T}} R_0^{-1}, \\
\beta_1 &= R_1 R_0^{-1}, & \pi_1 &= R_0 - \beta_1 R_1^{\mathrm{T}}, \qquad (5.39) \\
\Sigma_1 &= R_0 - \Delta_1 R_1, & Q_1 &= \Lambda_0 - \theta_1 S_1^{\mathrm{T}}.
\end{aligned}
$$

AL dedus anterior este prezentat sintetic în Tabelul 5.1. care indică şi numărul de operaţii aritmetice necesitate de implementarea fiecărui pas al algoritmului. Se observă din tabel că trecerea de la soluţia de ordin n la cea de ordin $(n+1)$ necesită $5n \cdot \mathcal{O}(m^3)$[flop] (presupunând că $n \gg 1$), unde $m = \max(m_u, m_y)$.

Rezultă că numărul total de operaţii aritmetice efectuate de AL pentru determinarea $\{\theta_n, Q_n\}$, $(n = 1, \ldots, n_{max})$ este, în mod aproximativ, următorul:

$$
2,5n_{max}^2 \cdot \mathcal{O}(m^3)\ [flop]. \qquad (5.40)
$$

Deci AL este, într-adevăr, cu două ordine de mărime mai rapid decât un algoritm standard care nu exploatează structura specială a problemei în studiu.

În condiţii generale asupra semnalului de intrare $\{u(t)\}$, matricea Γ_n, construită din covarianţele teoretice (5.5) sau covarianţele de eşantion (5.14), va fi pozitiv definită. Dacă $\Gamma_n > 0$, atunci $\Gamma_1, \Gamma_2, \ldots, \Gamma_{n-1}$ sunt, de asemenea, matrice pozitiv definite. Din proprietăţile matricelor pozitiv definite partiţionate, rezultă atunci că $\Sigma_n > 0$ şi $\pi_n > 0$ $(n = 1, 2, \ldots)$. Concluzia este că AL va lucra "normal" în cazurile în care $\Gamma_n > 0$.

Privind alte utilizări posibile ale AL dedus mai sus, în afara celei principale constând în calculul recursiv-în-ordin al secvenţei de ponderare, să facem următoarele observaţii:

a) Este relativ simplu de observat că perechile de variabile $\{\Delta_n, \Sigma_n\}$ şi $\{\beta_n, \pi_n\}$ determină două tipuri de factorizări triunghiulare ale matricei inverse de covarianţă, Γ_{n+1}^{-1}. Nu vom insista asupra detaliilor acestor factorizări care sunt perfect similare celor descrise în Lema 4.2.1..

b) Observaţia de mai sus sugerează că fie $\{\pi_n\}$ fie $\{\Sigma_n\}$ $(n = 1, 2, \ldots)$ pot fi utilizate pentru a testa pozitivitatea lui Γ_n. Această utilizare posibilă a AL este perfect similară celei a W²R descrisă în Capitolul precedent (bazată pe Lema 4.2.2.). Din nou nu vom intra în detalii deoarece nu am avea nimic nou de relevat faţă de cele spuse în Capitolul 4.

c) În fine, să notăm că modelul "sumă de convoluţie trunchiată", (5.2), este totdeauna stabil şi deci problema studiului stabilităţii nu are sens în acest caz.

Scanned with OKEN Scanner

Ecuaţie	Număr operaţii [flop]
Iniţializare:	
$\theta_1 = S_1 R_0^{-1} \quad \Delta_1 = R_1^{\mathrm{T}} R_0^{-1}$	$\mathcal{O}(m^3)$
$\beta_1 = R_1 R_0^{-1} \quad \pi_1 = R_0 - \beta_1 R_1^{\mathrm{T}}$	$\mathcal{O}(m^3)$
$\Sigma_1 = R_0 - \Delta_1 R_1 \quad Q_1 = \Lambda_0 - \theta_1 S_1^{\mathrm{T}}$	$\mathcal{O}(m^3)$
Pentru $n = 1, \ldots, n_{max}$ calculează:	
$P_n = R_{-(n+1)} - \Delta_n \phi_n = (R_{n+1} - \beta_n \psi_n)^{\mathrm{T}}$	$n \cdot \mathcal{O}(m^3)$
$\Omega_n = S_{n+1} - \theta_n \psi_n$	$n \cdot \mathcal{O}(m^3)$
$K_{n+1} = \Omega_n \Sigma_n^{-1}$	$\mathcal{O}(m^3)$
$\overline{K}_{n+1} = P_n \pi_n^{-1}$	$\mathcal{O}(m^3)$
$Q_{n+1} = Q_n - K_{n+1} \Omega_n^{\mathrm{T}}$	$\mathcal{O}(m^3)$
$\Sigma_{n+1} = \Sigma_n - \overline{K}_{n+1} P_n^{\mathrm{T}}$	$\mathcal{O}(m^3)$
$\pi_{n+1} = \pi_n - \bar{K}_{n+1} P_n$	$\mathcal{O}(m^3)$
$\theta_{n+1} = \begin{bmatrix} \theta_n & 0 \end{bmatrix} + K_{n+1} \begin{bmatrix} -\Delta_n & I \end{bmatrix}$	$\sim n \cdot \mathcal{O}(m^3)$
$\Delta_{n+1} = \begin{bmatrix} 0 & \Delta_n \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} I & -\beta_n \end{bmatrix}$	$\sim n \cdot \mathcal{O}(m^3)$
$\beta_{n+1} = \begin{bmatrix} \beta_n & 0 \end{bmatrix} + \overline{K}_{n+1} \begin{bmatrix} -\Delta_n & I \end{bmatrix}$	$\sim n \cdot \mathcal{O}(m^3)$

Tabelul 5.1. Algoritmul Levinson pentru modele multivariabile de tip sumă de convoluţie.

Scanned with OKEN Scanner

Scanned with OKEN Scanner

Capitolul 6

Algoritm Rapid pentru Modele Multivariabile de Tip Ecuaţie cu Diferenţe

6.1. Modele de tip ecuaţie cu diferenţe

Forma generală a unei ecuaţii cu diferenţe asociate unui sistem cu intrare $u(t)$ (m_u–dimensională) şi ieşire $y(t)$ (m_y – dimensională), este următoarea:

$$C(q^{-1})y(t) = D(q^{-1})u(t) + v(t), \tag{6.1}$$

unde q^{-1} este operatorul de întârziere cu un tact, $C(q^{-1})$ şi $D(q^{-1})$ sunt polinoame matriceale în q^{-1},

$$\begin{array}{rcll} C(q^{-1}) & = & I + C_1 q^{-1} + \ldots + C_n q^{-n} & (m_y \times m_y), \\ D(q^{-1}) & = & D_1 q^{-1} + \ldots + D_n q^{-n} & (m_y \times m_u), \end{array}$$

iar $v(t)$ este un "zgomot de ecuaţie" (m_y–dimensional). Ecuaţiile cu diferenţe de forma lui (6.1) au aplicaţii similare modelelor de tip sumă de convoluţie considerate în capitolul anterior. Totuşi, spre deosebire de acestea din urmă, ecuaţiile cu diferenţe sunt reprezentări mult mai economicoase din punctul de vedere al numărului de parametri implicaţi. Spre exemplu, un sistem liniar de ordinul întâi poate fi descris printr-o ecuaţie cu diferenţe cu $n = 1$, dar descrierea lui printr-o sumă de convoluţie poate necesita sute de parametri Markov $\{G_k\}$ (vezi (5.1)).

Problema de modelare corespunzătoare ecuaţiei (6.1) constă în determinarea coeficienţilor matriceali $\{C_i, D_j\}$ din măsurători intrare–ieşire (toate elementele acestor coeficienţi matriceali sunt presupuse necunoscute).

Putem utiliza metoda CMMP pentru rezolvarea acestei probleme de modelare. În cadrul acestei metode parametrii $\{C_i, D_j\}$ sunt obţinuţi prin minimizarea fie a matricei de covarianţă a lui $v(t)$, fie a unei funcţii monoton crescătoare de această matrice. De exemplu, putem determina $\{C_i, D_j\}$ minimizând urma matricei de covarianţă a lui $v(t)$:

$$Ev^{\mathrm{T}}(t)v(t) = E\|C(q^{-1})y(t) - D(q^{-1})u(t)\|^2, \tag{6.2}$$

unde $\|\cdot\|$ denotă norma euclidiană.

Problema de minimizare (6.2) conduce la o problemă numerică cu o structură specială care poate fi exploatată pentru a obţine algoritmi rapizi de rezolvare. În continuare, totuşi, nu vom trata problema (6.2) ci o problemă mai generală. Aşa cum vom vedea, rezolvarea problemei generalizate se poate face utilizând algoritmi rapizi introduşi în capitolele anterioare.

În multe aplicaţii practice ne interesează nu numai studiul dinamicii legăturii directe $u(t) \to y(t)$, ci şi studiul unei posibile legături inverse $y(t) \to u(t)$, pentru care putem considera următorul model similar cu (6.1):

$$F(q^{-1})u(t) = G(q^{-1})y(t) + w(t). \tag{6.3}$$

În (6.3), $w(t)$ este un zgomot (m_u-dimensional), $F(q^{-1})$ şi $G(q^{-1})$ sunt următoarele polinoame în q^{-1}:

$$\begin{aligned} F(q^{-1}) &= I + F_1 q^{-1} + \ldots + F_n q^{-n} && (m_u \times m_u), \\ G(q^{-1}) &= G_1 q^{-1} + \ldots + G_n q^{-n} && (m_u \times m_y). \end{aligned}$$

Dacă legătura inversă (6.3) există în mod real, atunci determinarea parametrilor $\{G_i, F_j\}$ ce o caracterizează este o problemă practic interesantă. Dacă legătura inversă $y(t) \to u(t)$ nu există, modelul (6.3) cu $G_i = F_i = 0 \quad (i = 1, \ldots, n)$ rămâne, totuşi, o descriere validă şi estimarea parametrilor lui (în presupunerea $n \geq 1$) poate fi considerată ca un *test asupra existenţei sau inexistenţei reacţiei*. Într-adevăr, dacă parametrii estimaţi $\{F_1, G_1, \ldots\}$ au valori nesemnificative (adică aproximativ egale cu zero), atunci putem conchide că nu există reacţie; în schimb dacă valorile estimate ale acestor parametri sunt semnificative, atunci există reacţia a cărui dinamică este modelată de ecuaţia de forma (6.3) estimată din măsurători $\{u(t), y(t)\}$.

Într-un context oarecum legat de discuţia anterioară, să facem următoarea observaţie: presupunerea că gradele polinoamelor C, D, F şi G sunt toate egale cu n nu este restrictivă; dacă aceste polinoame au grade diferite n_c, n_d, n_f şi n_g, atunci definind $n = \max(n_c, n_d, n_f, n_g)$ şi introducând anumiţi coeficienţi egali cu zero, putem considera că gradul tuturor polinoamelor este n.

Parametrii $\{F_i, G_j\}$ ai modelului legăturii inverse, (6.3), pot fi determinaţi tot prin metoda CMMP, adică prin minimizarea următoarei funcţii criteriu (comparaţi cu (6.2)):

$$Ew^{\mathrm{T}}(t)w(t) = E\|F(q^{-1})u(t) - G(q^{-1})y(t)\|^2. \tag{6.4}$$

Făcând presupunerea naturală că modelele (6.1) şi (6.3) nu au parametri comuni, putem combina (6.2) şi (6.4) într-un singur criteriu. Pentru a fi mai exacţi, să introducem următoarele notaţii:

$$\varepsilon(t) = \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}, \tag{6.5}$$

$$z(t) = \begin{bmatrix} y(t) \\ u(t) \end{bmatrix}, \tag{6.6}$$

$$A_i = \begin{bmatrix} C_i & -D_i \\ -G_i & F_i \end{bmatrix} \quad (i = 1, \ldots, n), \tag{6.7}$$

$$A(q^{-1}) = I + A_1 q^{-1} + \ldots + A_n q^{-n}. \tag{6.8}$$

Utilizând aceste notaţii putem scrie ecuaţiile cu diferenţe (6.1) şi (6.3) sub formă compactă

$$A(q^{-1})z(t) = \varepsilon(t). \tag{6.9}$$

Scanned with OKEN Scanner

Mai mult, să observăm că

$$E\varepsilon^{\mathrm{T}}(t)\varepsilon(t) = Ev^{\mathrm{T}}(t)v(t) + Ew^{\mathrm{T}}(t)w(t). \tag{6.10}$$

Deoarece cei doi termeni în (6.10) nu depind de parametri comuni, minimizarea criteriilor (6.2) în raport cu $\{C_i, D_j\}$ şi (6.4) în raport cu $\{F_i, G_j\}$ este echivalentă cu minimizarea lui (6.10) în raport cu $\{A_i\}$. Rezultă din discuţia anterioară că problema determinării parametrilor ecuaţiilor cu diferenţe (6.2) şi (6.4) poate fi *reformulată ca o problemă de modelare AR multivariabilă* (6.5)-(6.10).

Ca o consecinţă imediată a acestei reformulări, putem utiliza *algoritmul* W^2R pentru calculul rapid al estimaţiilor CMMP ale parametrilor $\{C_i, D_j, F_k, G_p\}$. Stabilitatea modelului astfel obţinut este studiată în paragraful următor.

6.2. Studiul stabilităţii modelului

Modelul (6.1) obţinut prin minimizarea criteriului CMMP (6.2) *nu* este în mod necesar stabil. Totuşi stabilitatea acestui model poate fi garantată prin alegerea corespunzătoare a condiţiilor experimentale (când acest lucru este posibil). Avem în acest sens următorul rezultat.

Teorema 6.2.1. *Să presupunem că sistemul în studiu este cauzal (astfel,* $y(t)$ *depinde de* $u(t), u(t-1)$ *etc.); mai mult, să presupunem că intrarea* $u(t)$ *este un* semnal alb. *Atunci modelul CMMP (6.1), (6.2) este stabil:*

$$\det[z^n C(z^{-1})] \neq 0 \quad \textit{pentru } |z| \geq 1. \tag{6.11}$$

Demonstraţie: Polinoamele $C(q^{-1})$ şi $D(q^{-1})$ care minimizează criteriul CMMP (6.2) sunt identice cu cele care minimizează criteriul extins (6.10) (vezi discuţia din jurul ecuaţiei (6.10)). Cealaltă pereche de polinoame $\{G(q^{-1}), F(q^{-1})\}$ care minimizează (6.10) se poate determina în următorul mod. Să observăm întâi că

$$\begin{aligned} w(t) &= F(q^{-1})u(t) - G(q^{-1})y(t) = \\ &= u(t) + \begin{bmatrix} F_1 & \dots & F_n & G_1 & \dots & G_n \end{bmatrix} \begin{bmatrix} u(t-1) \\ \vdots \\ u(t-n) \\ -y(t-1) \\ \vdots \\ -y(t-n) \end{bmatrix} \triangleq \\ &\triangleq u(t) + \beta\phi(t). \end{aligned} \tag{6.12}$$

Estimaţia CMMP a matricei β are următoarea expresie (vezi, de exemplu, (4.8)):

$$\beta = -\{Eu(t)\phi^{\mathrm{T}}(t)\}\{E\phi(t)\phi^{\mathrm{T}}(t)\}^{-1}. \tag{6.13}$$

Totuşi, făcând uz de presupunerile că sistemul este cauzal (deci $\phi(t)$ depinde de $u(t-1), u(t-2)$ ş.a.m.d) şi că $u(t)$ este semnal alb (astfel $Eu(t)u(s) = 0$ pentru $t \neq s$), rezultă că

$$Eu(t)\phi^{\mathrm{T}}(t) = 0$$

Scanned with OKEN Scanner

şi, prin urmare, $\beta = 0$. Astfel polinoamele F şi G care minimizează criteriul generalizat (6.10) sunt următoarele:

$$F(q^{-1}) \equiv I, \quad G(q^{-1}) \equiv 0. \tag{6.14}$$

Polinomul matriceal $A(q^{-1})$ care minimizează (6.10) este stabil (vezi Teorema 4.3.1.):

$$\det[z^n A(z^{-1})] \neq \text{ pentru } |z| \geq 1. \tag{6.15}$$

Totuşi, urmează din (6.14) şi discuţia anterioară că $A(q^{-1})$care minimizează (6.10) are următoarea structură,

$$A(q^{-1}) = \begin{bmatrix} C(q^{-1}) & D(q^{-1}) \\ 0 & I \end{bmatrix}. \tag{6.16}$$

Astfel,

$$\det[z^n A(z^{-1})] = \det[z^n C(z^{-1})] \tag{6.17}$$

care, împreună cu (6.15), implică (6.11).

□

În experimentele active de identificare, în cadrul cărora semnalul de intrare este generat de experimentator, satisfacerea condiţiilor Teoremei 6.1 nu ridică probleme. Totuşi, în experimentele pasive de identificare, semnalul de intrare nu mai este la îndemâna noastră şi el poate să nu fie alb. În astfel de cazuri modelul CMMP (6.1) poate să fie instabil (în ciuda faptului că sistemul identificat este stabil!)[73].

În fine, să mai notăm că, în condiţiile Teoremei 6.2.1., este posibilă deducerea unor *limite subunitare* pe modulul rădăcinilor lui $\det[z^n C(z^{-1})]$, limite similare cu cele prezentate în Capitolul 1 pentru modele AR monovariabile (vezi Teorema 3.2.1.). Cititorul interesat în deducerea şi utilizarea acestor limite subunitare poate consulta [83].

Scanned with OKEN Scanner

Scanned with OKEN Scanner

Partea a doua

Metode Rapide Recursive în Ordinul Modelului şi în Timp

Această parte a lucrării prezintă algoritmi rapizi recursivi atât în ordinul modelului, cât şi în timp, pentru estimarea parametrilor următoarelor tipuri de modele: modele "autoregresive şi de medie alunnecătoare" (ARMA); modele "sumă de convoluţie"; modele "ecuaţie cu diferenţe".

Aceşti algoritmi rapizi sunt de tip on-line, determinând la fiecare moment de eşantionare parametrii tuturor modelelor de ordine de la unu până la un ordin maxim preimpus, N, cu un efort de calcul $\mathcal{O}(N^2)$ la fiecare pas de eşantionare. În cazul în care numai semnalul de la ieşirea modelului este necesar (parametrii modelului nu sunt explicit ceruţi), calculele pot fi foarte eficient organizate utilizând structuri speciale de filtre, numite structuri latice, necestitând $\mathcal{O}(N)$ calcule la fiecare pas de eşantionare.

Scanned with OKEN Scanner

Capitolul 7

Algoritmul Latice pentru Modele AR Monovariabile

7.1. Parametrizarea latice a modelelor AR

Unul dintre cele mai importante domenii de utilizare ale modelelor AR este domeniul predicţiei liniare, acest fapt explicând utilizarea frecventă în descrierea metodelor rapide a unor termeni specifici ca: predicţie, predictor, eroare de predicţie, care vor fi definiţi mai jos. Prin intermediul acestor termeni se poate realiza o imagine intuitivă a structurii şi semnificaţiei mărimilor care apar în algoritmii rapizi, imagine necesară mai ales datorită complexităţii destul de ridicate a acestor algoritmi.

Fiind dată seria de timp $\{y(t)\}$, se defineşte *predicţia* de ordin n peste k paşi, notată $\hat{y}_n(t+k|t)$, ca fiind combinaţia liniară a celor mai recente n date disponibile la momentul t:

$$\hat{y}_n(t+k|t) = -a_1 y(t) - a_2 y(t-1) - \ldots - a_n y(t-n+1). \tag{7.1}$$

Vectorul format din coeficienţii combinaţiei liniare poartă numele de *predictor*, iar valorile succesive ale seriei utilizate în calculul predicţiei definesc vectorul observaţiilor

$$\varphi_n(t) = \left[\begin{array}{cccc} -y(t-1) & -y(t-2) & \ldots & -y(t-n) \end{array} \right]^{\mathrm{T}}. \tag{7.2}$$

Predicţia peste un singur pas se mai numeşte predicţie înainte:

$$\begin{aligned} \hat{y}_n(t) &\triangleq \hat{y}_n(t|t-1) = -a_1 y(t-1) - a_2 y(t-2) - \ldots - a_n y(t-n) = \\ &= \theta_n^{\mathrm{T}} \varphi_n(t), \end{aligned} \tag{7.3}$$

iar vectorul coeficienţilor combinaţiei liniare,

$$\theta_n = \left[\begin{array}{cccc} a_1 & a_2 & \ldots & a_n \end{array} \right]^{\mathrm{T}}, \tag{7.4}$$

reprezintă predictorul înainte de ordin n. Eroarea de predicţie înainte de ordin n se defineşte prin relaţia

$$\begin{aligned} \varepsilon_n^f(t) &= y(t) - \hat{y}_n(t) = y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = \\ &= y(t) - \theta_n^{\mathrm{T}} \varphi_n(t). \end{aligned} \tag{7.5}$$

Predicţia de ordin n peste $k = -n$ paşi se numeşte predicţie înapoi

$$\hat{y}_n^b(t-n) \triangleq \hat{y}_n(t-n|t,\ldots,t-n+1) = -b_1 y(t) - b_2 y(t-1) - \ldots - b_n y(t-n+1) = \\ = \theta_n^{b\mathrm{T}} \varphi_n(t+1) \tag{7.6}$$

şi reprezintă de fapt extrapolarea în trecut, printr-o combinaţie liniară, a celui mai recent segment de n observaţii succesive ale seriei. Vectorul notat

$$\theta_n^b = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix}^{\mathrm{T}}$$

se numeşte predictorul înapoi de ordin n, iar mărimea

$$\varepsilon_n^b(t) = y(t-n-1) - \hat{y}_n^b(t-n-1) = y(t-n-1) + b_1 y(t-1) + \ldots + b_n y(t-n) = \\ = y(t-n-1) - \theta_n^{b\mathrm{T}} \varphi_n(t) \tag{7.7}$$

este eroarea de predicţie înapoi, corespunzătoare predictorului înapoi de ordin n, θ_n^b şi vectorului observaţiilor $\varphi_n(t)$.

După această introducere în terminologia domeniului predicţiei liniare, vom formula o problemă de predicţie a cărei soluţie eficientă numeric va fi prezentată în această secţiune.

Problema 7.1.1. *Fiind dată seria staţionară $\{y(t)\}$, să se găsească predicţiile înainte optimale (de dispersie minimă) $\hat{y}_n(N)$ şi predicţiile înapoi optimale $\hat{y}_n^b(N-n-1)$ pentru toate ordinele n, cu $1 \leq n \leq n_{max}$, printr-o metodă recursivă în timp.*

Soluţia generală a problemei de predicţie optimală se obţine cu ajutorul lemei următoare, în care vor fi folosite notaţiile (1.7),(1.8),(1.9).

Lema 7.1.1. *Fie seria $\{y(t)\}$ staţionară. Dacă matricea R este nesingulară, atunci:*

i) predictorul înainte optimal este dat de

$$\theta_n = -R^{-1} r, \tag{7.8}$$

iar dispersia erorii de predicţie înainte optimală este

$$\sigma_n \triangleq E\varepsilon_n^{f2}(t) = \rho_0 + r^{\mathrm{T}} \theta_n. \tag{7.9}$$

ii) predictorul înapoi optimal este

$$\theta_n^b = -R^{-1} \tilde{r} = \tilde{\theta}_n, \tag{7.10}$$

iar dispersia erorii de predicţie înapoi optimală este

$$\sigma_n^b \triangleq E\varepsilon_n^{b2}(t) = \sigma_n. \tag{7.11}$$

Demonstraţie. Pentru demonstrarea părţii (i), să observăm că de fapt problema predicţiei înainte optimale este o simplă reformulare a problemei de modelare AR tratată în Capitolul 1. În consecinţă, predictorul înainte optimal va fi dat de (1.13), iar dispersia optimală va fi dată de (1.14).

Scanned with OKEN Scanner

Pentru a demonstra partea (ii) a lemei, vom rescrie dispersia erorii de predicţie înapoi astfel:

$$\begin{aligned} E\varepsilon_n^{b2}(t) &= E[y(t-n-1) - \theta_n^{bT}\varphi_n(t)]^2 = \\ &= Ey^2(t-n-1) + 2\theta_n^{bT} E \begin{bmatrix} y(t-1) \\ \cdot \\ \cdot \\ \cdot \\ y(t-n) \end{bmatrix} y(t-n-1) + \theta_n^{bT} E\varphi_n(t)\varphi_n^T(t)\theta_n^b = \\ &= \rho_0 + 2\theta_n^{bT}\tilde{r} + \theta_n^{bT} R\theta_n^b = \\ &= (\theta_n^b + R^{-1}\tilde{r})^T R(\theta_n^b + R^{-1}\tilde{r}) + \rho_0 - \tilde{r}^T R^{-1}\tilde{r}. \end{aligned} \tag{7.12}$$

Din (1.12) şi datorită nesingularităţii matricei R, urmează că R este pozitiv definită şi minimul dispersiei erorii de predicţie înapoi de ordin n se obţine pentru predictorul optimal (7.10), iar dispersia erorii de predicţie înapoi optimale este

$$\sigma_n^b = \rho_0 - \tilde{r}^T R^{-1}\tilde{r} = \rho_0 + \tilde{r}^T\tilde{\theta}_n = \rho_0 + r^T\theta_n = \sigma_n. \tag{7.13}$$

□

O modalitate de rezolvare a Problemei 7.1.1. este următoarea: se estimează covarianţele de eşantion (de exemplu prin (1.15)), care determină matricea R şi vectorul r, apoi se rezolvă sistemul (7.8), iar valoarea predictorului optimal se înlocuieşte în relaţiile (7.3) şi (7.6) rezultând predicţiile optimale.

Calculul predictorului optimal (7.8) se poate realiza foarte eficient numeric utilizând ALD. Totuşi, estimarea covarianţelor de eşantion prin (1.15) implică un mare volum de calcule. Dacă Problema 7.1.1. trebuie rezolvată pentru toate momentele de timp $N = 1, 2, \ldots$ (şi aceasta este o cerinţă impusă în foarte multe aplicaţii), atunci este mult mai raţional să evaluăm în mod recursiv covarianţele de eşantion utilizând una din formulele

$$\rho_k(N) = \frac{N-1}{N}\rho_k(N-1) + \frac{1}{N}y(N)y(N-k) = \rho_k(N-1) + \frac{1}{N}(y(N)y(N-k) - \rho_k(N-1)). \tag{7.14}$$

Introducând varianta scalată a covarianţelor $\rho_k(N)$

$$\rho_k'(N) = N\rho_k(N) = \sum_{t=k+1}^{N} y(t)y(t-k) = \rho_k'(N-1) + y(N)y(N-k), \tag{7.15}$$

rezultă o reducere a numărului de calcule faţă de (7.14) şi se observă din (7.8), (1.8) şi (1.9) că această scalare nu influenţează valoarea predictorului optimal.

În Tabelul 7.1. este sintetizată prima soluţie a Problemei 7.1.1., utilizând ALD.

Iniţializarea la $N = 0$ cu valori nule pentru toate "covarianţele scalate" face ca matricea R să fie singulară pe perioada primelor $0 \leq N \leq N_0$ momente de timp ($N_0 > n_{max}$) şi deci secţiunea ALD nu poate fi iterată. Această secţiune va putea fi utilizată numai după ce numărul de date disponibile a devenit mai mare ca $N_0 > n_{max}$ (de obicei, datorită impreciziei estimaţiilor obţinute cu N mic, secţiunea ALD este iterată numai pentru valori "mari" ale lui N, deci se alege $N_0 \gg n_{max}$). Se observă că la fiecare moment de timp N este necesar un număr total de operaţii $\mathcal{O}(n_{max}^2)$ pentru a calcula atât predicţiile optimale, cât şi predictorii optimali.

În continuare vom arăta că, dacă valorile predictorilor nu sunt explicit cerute (deci în contextul Problemei 7.1.1.), vom putea evita utilizarea paşilor cei mai costisitori numeric din algoritmul Levinson, astfel încât Problema 7.1.1. să poată fi rezolvată în $\mathcal{O}(n_{max})$ operaţii.

Scanned with OKEN Scanner

Algoritmul 7.1.1.

ALD RECURSIV ÎN TIMP		*Număr operaţii/ ecuaţie* ⋆ / +	*Număr operaţii/ pas alg.*
1. *Iniţializarea recursiilor în timp* $(N = 0)$			
Pentru $n = 0, \ldots, n_{max}$			
$\rho'_n(0) = 0$			
2. *Recursii în timp*			$\mathcal{O}(n^2_{max})$
Pentru $N = 1, \ldots, N_{max}$			
2.1 *Pentru* $n = 0, \ldots, n_{max}$			$\mathcal{O}(n_{max})$
$\rho'_n(N) = \rho'_n(N-1) + y(N)y(N-k)$	(1)	1 0 1	
2.2 *Dacă* $N \geq N_0$			
Atunci			
i. Iniţializare recursii în ordin			$\mathcal{O}(1)$
$\theta_1 = -\rho'_1(N)/\rho'_0(N) = K_1$	(2)	0 1 0	
$\sigma'_1 = -\rho'_0(N) - \rho'^2_1(N)/\rho'_0(N)$	(3)	0 0 1	
$\hat{y}^f_1(N) = -\theta_1 y(N-1)$	(4)	1 0 0	
$\hat{y}^b_1(N-2) = -\theta_1 y(N-1)$	(5)	1 0 0	
ii. Recursii în ordin			$\mathcal{O}(n^2_{max})$
Pentru $n = 1, \ldots, n_{max} - 1$			
$K_{n+1} = -\frac{\rho'_{n+1}(N) + \theta^T_n \tilde{r}'_n(N)}{\sigma'_n}$	(6)	n 1 n	
$\sigma'_{n+1} = \sigma'_n(1 - K^2_{n+1})$	(7)	2 0 1	
$\theta_{n+1} = \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} + K_{n+1} \begin{bmatrix} \tilde{\theta}_n \\ 1 \end{bmatrix}$	(8)	n 0 n	
(Opt.) $\hat{y}_{n+1}(N) = \theta^T_{n+1}\varphi_{n+1}(N)$	(9)	n 0 n	
(Opt.) $\hat{y}^b_{n+1}(N-n-2) = \tilde{\theta}^T_{n+1}\varphi_{n+1}(N)$	(10)	n 0 n	

Tabelul 7.1. Algoritmul Levinson–Durbin recursiv în timp.

Scanned with OKEN Scanner

Mai întâi să observăm că erorile de predicţie înainte sunt legate de predicţiile înainte prin relaţia (7.5), pentru evaluarea căreia sunt necesare $\mathcal{O}(n_{max})$ operaţii (o observaţie similară se referă la erorile de predicţie înapoi şi predicţiile înapoi). Scopul în continuare este transformarea recurenţei predictorilor în recurenţa predicţiilor. Recurenţa predictorilor se obţine înlocuind (7.10) în (1.26),

$$\theta_{n+1} = \begin{bmatrix} \theta_n \\ 0 \end{bmatrix} + K_{n+1} \begin{bmatrix} \theta_n^b \\ 1 \end{bmatrix}, \tag{7.16}$$

iar prin reversarea vectorilor acestei ecuaţii rezultă

$$\theta_{n+1}^b = \begin{bmatrix} 0 \\ \theta_n^b \end{bmatrix} + K_{n+1} \begin{bmatrix} 1 \\ \theta_n \end{bmatrix}, \tag{7.17}$$

obţinându-se relaţii care exprimă recurenţa în ordin a predictorilor înainte şi înapoi. De aici rezultă relaţii recursive în ordin pentru eroarea de predicţie înainte (utilizând (7.16))

$$\begin{aligned}
\varepsilon_{n+1}^f(t) &= y(t) - \theta_{n+1}^{\mathrm{T}} \varphi_{n+1}(t) = \\
&= y(t) + \begin{bmatrix} \theta_n^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} y(t-1) \\ \vdots \\ y(t-n-1) \end{bmatrix} + K_{n+1} \begin{bmatrix} \theta_n^{b\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} y(t-1) \\ \vdots \\ y(t-n-1) \end{bmatrix} = \\
&= y(t) - \theta_n^{\mathrm{T}} \varphi_n(t) + K_{n+1}(y(t-n-1) - \theta_n^{b\mathrm{T}} \varphi_n(t)) = \varepsilon_n^f(t) + K_{n+1} \varepsilon_n^b(t),
\end{aligned} \tag{7.18}$$

şi perfect similar pentru eroarea de predicţie înapoi (utilizând (7.17))

$$\begin{aligned}
\varepsilon_{n+1}^b(t) &= y(t-n-2) - \theta_{n+1}^{b\mathrm{T}} \varphi_{n+1}(t) = \\
&= y(t-n-2) + \begin{bmatrix} 0 & \theta_n^{b\mathrm{T}} \end{bmatrix} \begin{bmatrix} y(t-1) \\ \vdots \\ y(t-n-1) \end{bmatrix} + K_{n+1} \begin{bmatrix} 1 & \theta_n^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} y(t-1) \\ \vdots \\ y(t-n-1) \end{bmatrix} = \\
&= y(t-n-2) - \theta_n^{b\mathrm{T}} \varphi_n(t-1) + K_{n+1}(y(t-1) - \theta_n^{\mathrm{T}} \varphi_n(t-1)) = \\
&= \varepsilon_n^b(t-1) + K_{n+1} \varepsilon_n^f(t-1).
\end{aligned} \tag{7.19}$$

Relaţiile (7.18), (7.19) utilizează pentru trecerea de la erorile de predicţie (echivalent, predicţiile) de ordin n la erorile de predicţie de ordin $n+1$, $\mathcal{O}(1)$ operaţii, înlocuind practic ultimele trei ecuaţii din Tabelul 7.1., ecuaţii ce necesitau $3n$ operaţii pentru fiecare iteraţie în ordin.

Pentru a înlocui şi ecuaţia (6) din algoritmul 7.1.1., singura care mai necesită n operaţii, trebuie găsită o altă modalitate de calcul a coeficienţilor de reflexie K_{n+1}, care depind de θ_n prin intermediul mărimii

$$\alpha_n = \rho_{n+1} + \theta_n^{\mathrm{T}} \tilde{r}_n. \tag{7.20}$$

Se poate arăta foarte simplu că α_n are o semnificaţie intuitivă, fiind egal cu intercovarianţa erorilor de predicţie optimale înainte şi înapoi de ordin n:

$$\begin{aligned}
E\varepsilon_n^f(t)\varepsilon_n^b(t) &= E(y(t) - \theta_n^{\mathrm{T}} \varphi_n(t))(y(t-n-1) - \theta_n^{b\mathrm{T}} \varphi_n(t)) = \\
&= \rho_{n+1} - \theta_n^{\mathrm{T}} E\varphi_n(t) y(t-n-1) - \theta_n^{b\mathrm{T}} E\varphi_n(t) y(t) + \theta_n^{\mathrm{T}} E\varphi_n(t)\varphi_n^{\mathrm{T}}(t)\theta_n^b = \\
&= \rho_{n+1} + \theta_n^{\mathrm{T}} \tilde{r}_n + \theta_n^{b\mathrm{T}} r_n + \theta_n^{\mathrm{T}} R \theta_n^b = \rho_{n+1} - r_n^{\mathrm{T}} R^{-1} \tilde{r}_n - \tilde{r}_n^{\mathrm{T}} R^{-1} r_n + r_n^{\mathrm{T}} R^{-1} R R^{-1} \tilde{r}_n \\
&= \rho_{n+1} - r_n^{\mathrm{T}} R^{-1} \tilde{r}_n = \rho_{n+1} + \theta_n^{\mathrm{T}} \tilde{r}_n = \alpha_n.
\end{aligned} \tag{7.21}$$

Scanned with OKEN Scanner

Rezultă imediat că o altă modalitate de calcul a mărimii α_n ar fi estimarea intercovarianţei teoretice (7.21) prin intercovarianţa de eşantion

$$\alpha_n(N) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_n^f(t)\varepsilon_n^b(t), \tag{7.22}$$

însă pentru simplificarea evaluării recursive în timp introducem mărimea

$$\alpha_n'(N) = N\alpha_n(N) = \alpha_n'(N-1) + \varepsilon_n^f(N)\varepsilon_n^b(N). \tag{7.23}$$

În fine, deşi pentru dispersia erorilor de predicţie optimală $\sigma_n(N)$ există relaţii recursive în ordin eficiente ($\mathcal{O}(1)$ operaţii), pentru uniformitatea procedurii de iniţializare în ordin a noului algoritm va fi preferată o relaţie recursivă în timp şi pentru σ_n. Astfel, din (7.9) rezultă estimaţia

$$\sigma_n(N) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_n^{f2}(t), \tag{7.24}$$

sau, introducând varianta scalată σ_n', rezultă

$$\sigma_n'(N) = N\sigma_n(N) = \sigma_n'(N-1) + \varepsilon_n^{f2}(N). \tag{7.25}$$

Se observă că efectele de scalare din (7.23) şi (7.25) se compensează

$$K_{n+1}(N) = -\frac{\alpha_n(N)}{\sigma_n(N)} = -\frac{\alpha_n'(N)}{\sigma_n'(N)}. \tag{7.26}$$

Sistematizând relaţiile deduse mai sus se obţine corpul recursiilor în ordin (3)-(9) din algoritmul 7.1.2.

Pentru iniţializarea acestor recursii vom introduce ordinul fictiv "0" şi vom adopta

$$\varepsilon_0^f(N) = y(N), \quad \varepsilon_0^b(N) = y(N-1), \tag{7.27}$$

rezultând după prima executare a recursiilor în ordin (3)-(8):

$$\begin{aligned}
\sigma_0'(N) &= \sum_{t=1}^{N} y^2(t) = \rho_0'(N),\\
\alpha_0'(N) &= \sum_{t=1}^{N} y(t)y(t-1) = N\rho_1(N),\\
K_1(N) &= -\rho_1'(N)/\rho_0'(N) = \theta_1,\\
\hat{y}_1(N) &= -\theta_1 y(N-1),
\end{aligned}$$

deci exact valorile cu care ar fi trebuit iniţializat algoritmul pentru ordinul $n = 1$. Iniţializarea la $n = 0$, deşi "fictivă", conduce la o formă mai concisă a algoritmului.

Iniţializarea recursiei în timp, necesară la momentul N_0 de start al algoritmului, poate fi realizată prin rularea Algoritmului 7.1.1. (până la momentul $N_{max} = N_0$) care va realiza calculul covarianţelor $\{\rho_k(N_0)\}_{k=1}^{n_{max}}$ şi apoi vor rezulta valorile la momentul N_0 pentru mărimile de interes: $\alpha_n', \sigma_n', \varepsilon_n^f, \varepsilon_n^b$. Această iniţializare necesită $\mathcal{O}(n_{max}^2)$ operaţii, dar în continuare, la fiecare moment de timp $N > N_0$, algoritmul 7.1.2. se execută în $\mathcal{O}(n_{max})$ operaţii.

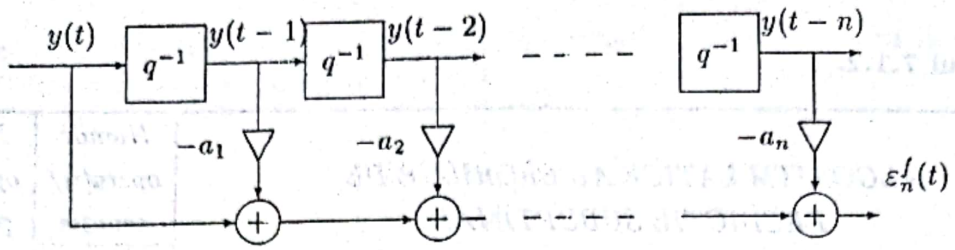
Scanned with OKEN Scanner

Algoritmul 7.1.2.

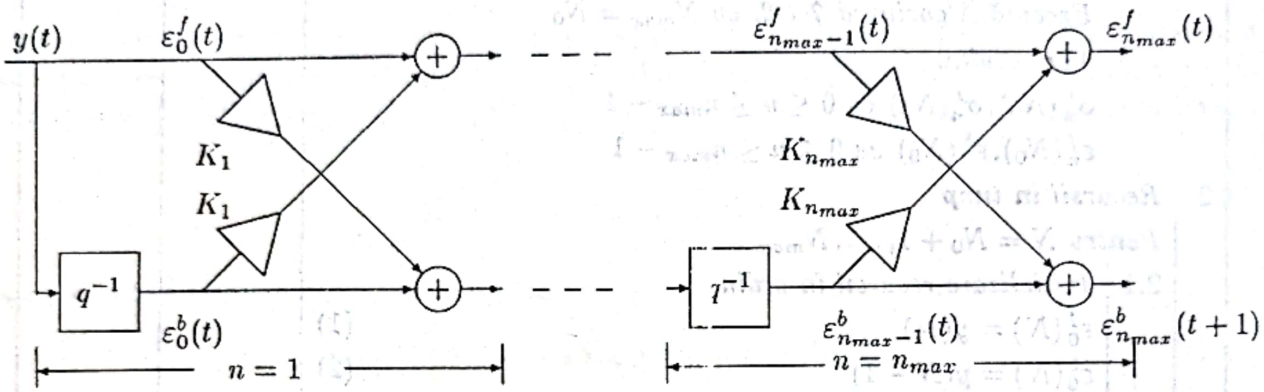
ALGORITM LATICE AL ERORILOR DE PREDICTIE SUBOPTIMAL		*Număr operaţii/ ecuaţie* ⋆ / +			*Număr operaţii/ pas alg.*
1. *Iniţializarea recursiilor în timp la* $(N = N_0)$					$\mathcal{O}(n_{max}^2)$
Execută Algoritmul 7.1.2. cu $N_{max} = N_0$					
Vor rezulta:					
$\alpha'_n(N_0), \sigma'_n(N_0)$ *cu* $0 \le n \le n_{max} - 1$					
$\varepsilon^f_n(N_0), \varepsilon^b_n(N_0)$ *cu* $0 \le n \le n_{max} - 1$					
2. *Recursii în timp*					
Pentru $N = N_0 + 1, \ldots, N_{max}$					
2.1 *Iniţializare recursii în ordin*					
$\varepsilon^f_0(N) = y(N)$	(1)				
$\varepsilon^b_0(N) = y(N-1)$	(2)				
2.2 *Recursii în ordin:*					$\mathcal{O}(n_{max})$
Pentru $n = 0, \ldots, n_{max} - 1$					
$\sigma'_n(N) = \sigma'_n(N-1) + \varepsilon^{f2}_n(N)$	(3)	1	0	1	
$\alpha'_n(N) = \alpha'_n(N-1) + \varepsilon^f_n(N)\varepsilon^b_n(N)$	(4)	1	0	1	
$K_{n+1}(N) = -\alpha'_n(N)/\sigma'_n(N)$	(5)	0	1	0	
$\varepsilon^f_{n+1}(N) = \varepsilon^f_n(N) + K_{n+1}(N)\varepsilon^b_n(N)$	(6)	1	0	1	
$\varepsilon^b_{n+1}(N) = \varepsilon^b_n(N-1) + K_{n+1}(N)\varepsilon^f_n(N-1)$	(7)	1	0	1	
(Opt.) $\hat{y}_{n+1}(N) = y(N) - \varepsilon^f_{n+1}(N)$	(8)	0	0	1	
(Opt.) $\hat{y}^b_{n+1}(N-n-2) = y(N-n-2) - \varepsilon^b_{n+1}(N)$	(9)	0	0	1	

Tabelul 7.2. Algoritmul latice al erorilor de predicţie suboptimal.

Scanned with OKEN Scanner



(a) Sistemul descris prin parametrizarea $\theta = [a_1\ a_2\ \ldots\ a_n]$



(b) Sistemul descris prin parametrizarea $\{K_i\}_{i=1}^{n}$

Figura 7.1. Structura sistemelor discrete descrise prin cele două tipuri de parametrizări: *(a)* $\theta = [a_1\ a_2\ \ldots\ a_n]^{\mathrm{T}}$ şi *(b)* $\{K_i\}_{i=1}^{n_{max}}$.

În algoritmul 7.1.2., renunţându-se la parametrizarea (7.4) a modelului AR (predictorului), predicţiile rezultă prin utilizarea unei reprezentări echivalente cu (7.4) a modelului AR şi anume prin utilizarea parametrilor K_n, numiţi şi coeficienţi de reflexie sau coeficienţi PARCOR (de corelaţie parţială), obţinându-se o eficienţă numerică mai bună cu un ordin de mărime.

La fiecare moment de timp se determină prin ecuaţiile (3),(4) şi (5) cei n_{max} parametri $\{K_n\}_{n=1}^{n_{max}}$ care reuşesc caracterizarea întregului set de modele AR optimale $\{\theta_n\}_{n=1}^{n_{max}}$, care prin (7.4) depind de $n_{max}^2/2$ parametri. Ecuaţiile (6),(7) arată de fapt cum poate fi utilizată parametrizarea în K_i pentru calculul erorilor de predicţie optimală.

Structura sistemului discret care este descris de parametrizarea (7.4) a modelului AR este reprezentată în Figura 7.1.*a*. În această parametrizare, trecerea de la modelul optimal de ordin n, cu parametrii θ_n, la modelul optimal de ordin $n+1$, cu parametrii θ_{n+1} se face prin schimbarea întregului set de parametri θ_n.

Structura sistemului discret care realizează dependenţa recursivă (7.18),(7.19) este reprezentată în Fig. 7.1.*b*. Pentru acest sistem, trecerea de la modelul optimal de ordin n, care conţine n blocuri, la modelul optimal de ordin $n+1$, se face prin păstrarea parametrilor primelor n blocuri, $K_1, \ldots, K_n$, şi adăugarea unui nou bloc caracterizat de parametrul K_{n+1}. Această structură poartă numele de structură latice şi este foarte utilizată la realizarea filtrelor numerice, dar şi în alte domenii în care s-a dovedit că este cea mai naturală modalitate de modelare, ca în seismică, sau modelarea tractului vocal.

La numeroasele utilizări ale parametrilor K_i în studierea stabilităţii sistemelor discrete [76]

Scanned with OKEN Scanner

se adaugă astfel şi construirea unor filtre foarte eficiente numeric pentru generarea predicţiilor optimale. În afară de rapiditate în rezolvarea problemei de predicţie optimală, filtrele latice prezintă şi avantajul unei sensibilităţi reduse la erorile de rotunjire rezultate din operaţiile aritmetice [16],[53], proprietate de o mare importanţă în implementările în virgulă fixă pe un număr mic de biţi. În legătură cu rezolvarea Problemei 7.1.1. putem face câteva observaţii:

1. Soluţia optimală dată în Lema 7.1.1. nu poate fi implementată exact decât dacă se cunosc covarianţele seriei staţionare. Dacă datele iniţiale sunt valorile seriei $\{y(t)\}_{t=1}^{N}$, atunci se poate obţine doar o implementare asimptotic optimală prin estimaţii consistente ale funcţiilor de covarianţă, ca în algoritmul 7.1.1. Şi algoritmul 7.1.2. este suboptimal, soluţia optimală putându-se obţine doar dacă s-ar cunoaşte valorile exacte ale coeficienţilor de reflexie. Datorită structurii neliniare a schemei de adaptare (estimare) a coeficienţilor de reflexie, convergenţa algoritmului 7.1.2. spre soluţia optimală este o problemă complexă, studiată în [23].

   În subcapitolele următoare se va renunţa la încercarea de a rezolva (aproximativ) probleme de optimizare cu criterii de tipul $Ee^2(t)$ şi se va încerca rezolvarea exactă a unor probleme cu criterii ceva mai puţin "tari", de tipul $\sum_{t=1}^{N} e^2(t)$. Performanţele ultimei abordări sunt superioare primei abordări, pentru cazul unui număr relativ mic de date disponibile. Pentru seturi mari de date, diferenţele în performanţe nu mai sunt atât de drastice, astfel încât algoritmii de tip 7.1.2. găsesc utilizări frecvente[27],[46],[47] şi [48].

2. Există mai multe variante ale Algoritmului 7.1.2., diferind prin modul de estimare a coeficienţilor de reflexie. Astfel, relaţiile

$$K_{n+1}^{1} = -\frac{E\varepsilon_n^f(t)\varepsilon_n^b(t)}{E\varepsilon_n^{f2}(t)} = -\frac{\alpha_n}{\sigma_n}, \tag{7.28}$$

$$K_{n+1}^{2} = -\frac{E\varepsilon_n^f(t)\varepsilon_n^b(t)}{E\varepsilon_n^{b2}(t)} = -\frac{\alpha_n}{\sigma_n^b}, \tag{7.29}$$

$$K_{n+1}^{3} = -\frac{E\varepsilon_n^f(t)\varepsilon_n^b(t)}{\sqrt{E\varepsilon_n^{f2}(t)E\varepsilon_n^{b2}(t)}} = -\frac{\alpha_n}{\sqrt{\sigma_n\sigma_n^b}}, \tag{7.30}$$

$$K_{n+1}^{4} = -\frac{E\varepsilon_n^f(t)\varepsilon_n^b(t)}{\frac{1}{2}\left[E\varepsilon_n^{f2}(t) + E\varepsilon_n^{b2}(t)\right]} = -\frac{2\alpha_n}{\sigma_n + \sigma_n^b} \tag{7.31}$$

sunt echivalente datorită relaţiei (7.11). Atunci când covarianţele şi intercovarianţele teoretice sunt înlocuite cu cele de eşantion, utilizând (7.22),(7.24) şi

$$\sigma_n^b = \frac{1}{N}\sum_{t=1}^{N} \varepsilon_n^{b2}(t),$$

estimaţiile rezultate pentru $K_{n+1}^{1}, \ldots, K_{n+1}^{4}$ vor fi diferite. Estimaţia lui K_n^3[46] are proprietatea că este mai mică în modul decât 1, ceea ce implică din criteriul Schur [76] stabilitatea sistemului discret din Figura 7.1.*b*. Estimaţia K_n^4[46] are proprietatea că minimizează $E(\varepsilon_{n+1}^{f2} + \varepsilon_{n+1}^{b2})$, şi este în general preferată în prelucrarea semnalelor în timp real, pentru că evită extragerea radicalului şi în acelaşi timp garantează stabilitatea filtrului recursiv latice.

Scanned with OKEN Scanner

7.2. Algoritmul "latice a erorilor de predicţie"

O clasă foarte bogată de algoritmi rapizi, care vor constitui obiectul capitolelor care urmează, sunt obţinuţi plecând de la metoda celor mai mici pătrate recursive în timp. Metoda constă în minimizarea recursivă a unui criteriu determinist, suma pătratelor erorilor de predicţie (de modelare) pe intervalul de observaţie.

În continuare se va formula problema de modelare care va face obiectul acestui subcapitol, se va prezenta soluţia problemei de optimizare cu criteriul CMMP determinist şi apoi, prin exploatarea structurii speciale a soluţiei, se va organiza calculul recursiv în ordin al predictorilor (asemănător cu ALD) şi al erorilor de predicţie; în final, se vor deduce ecuaţiile recursive în timp şi iniţializările necesare pentru a obţine un algoritm "latice a erorilor de predicţie".

Problema 7.2.1. *Fiind dată seria* $\{y(t)\}$, *să se găsească predicţiile înainte* $\{\hat{y}_n(N)\}_{n=1}^{n_{max}}$ *obţinute cu predictorii înainte care minimizează*

$$V_n(\theta) = \sum_{t=n}^{N}(y(t) - \hat{y}_n(t))^2 = \sum_{t=n}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta)^2 \tag{7.32}$$

şi predicţiile înapoi $\{\hat{y}_n^b(N-n-1)\}_{n=1}^{n_{max}}$ *obţinute cu predictorii înapoi care minimizează*

$$V_n^b(\theta^b) = \sum_{t=n}^{N}(y(t-n-1) - \hat{y}_n^b(t-n-1))^2 = \sum_{t=n}^{N}(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta^b)^2, \tag{7.33}$$

utilizând algoritmi recursivi în timp eficienţi numeric

Observaţii

1. Problema 7.2.1. este mai generală decât problema 7.1.1., întrucât a fost eliminată cerinţa restrictivă a staţionarităţii seriei $y(t)$.
2. Cuplarea problemei predicţiei înainte cu cea a predicţiei înapoi, deşi ultima apare mult mai rar practic, s-a dovedit deja avantajoasă în algoritmul 7.1.2., fiind cheia obţinerii unei eficienţe numerice ridicate.

Un prim rezultat necesar rezolvării problemei 7.2.1. este cuprins în următoarea lemă.

Lema 7.2.1. *Cunoscându-se valorile seriei* $\{y(t)\}_{t=-1}^{N}$, *notăm:*

$$R_n(N) = \sum_{t=n}^{N}\varphi_n(t)\varphi_n^{\mathrm{T}}(t), \tag{7.34}$$

$$r_n(N) = \sum_{t=n}^{N}\varphi_n(t)y(t), \tag{7.35}$$

$$r_n^b(N) = \sum_{t=n}^{N}\varphi_n(t)y(t-n-1), \tag{7.36}$$

$$\rho_0(N) = \sum_{t=n}^{N}y^2(t) \tag{7.37}$$

şi presupunem matricea $R_n(N)$ *nesingulară.*

Scanned with OKEN Scanner

i) Predictorul înainte care mimizează (7.32) este

$$\theta_n(N) = R_n^{-1}(N) r_n(N) \tag{7.38}$$

şi valoarea minimă a criteriului (7.32) este

$$\sigma_n(N) = \rho_0(N) - r_n^{\mathrm{T}}(N)\theta_n(N) \triangleq V_n(\theta_n(N)). \tag{7.39}$$

ii)Predictorul înapoi care mimizează (7.33) este

$$\theta_n^b(N) = R_n^{-1}(N) r_n^b(N) \tag{7.40}$$

şi valoarea minimă a criteriului (7.33) este

$$\sigma_n^b(N) = \rho_0(N-n-1) - r_n^{b\mathrm{T}}(N)\theta_n^b(N) \triangleq V_n^b(\theta_n^b(N)). \tag{7.41}$$

Demonstraţie. Criteriul (7.32) se poate scrie

$$\begin{aligned} V_n(\theta) &= \sum_{t=n}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta)^2 = \sum_{t=n}^{N} y^2(t) - 2\sum_{t=n}^{N} y(t)\varphi_n^{\mathrm{T}}(t)\theta + \theta^{\mathrm{T}} \sum_{t=n}^{N} \varphi_n(t)\varphi_n^{\mathrm{T}}(t)\theta = \\ &= \rho_0(N) - 2r_n^{\mathrm{T}}(N)\theta + \theta^{\mathrm{T}} R_n(N)\theta = \\ &= (\theta - R_n^{-1}(N)r_n(N))^{\mathrm{T}} R_n(N)(\theta - R_n^{-1}(N)r_n(N)) + \\ &\quad + \rho_0(N) - r_n^{\mathrm{T}}(N)R_n^{-1}r_n(N). \end{aligned} \tag{7.42}$$

În mod similar se poate transforma şi criteriul (7.33)

$$\begin{aligned} V_n^b(\theta^b) &= \sum_{t=n}^{N}(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta^b)^2 = \\ &= \sum_{t=n}^{N} y^2(t-n-1) - 2\sum_{t=n}^{N} y(t-n-1)\varphi_n^{\mathrm{T}}(t)\theta^b + \theta^{b\mathrm{T}} \sum_{t=n}^{N} \varphi_n(t)\varphi_n^{\mathrm{T}}(t)\theta^b = \\ &= (\theta^b - R_n^{-1}(N)r_n^b(N))^{\mathrm{T}} R_n(N)(\theta^b - R_n^{-1}(N)r_n^b(N)) + \\ &\quad + \rho_0(N-n-1) - r_n^{b\mathrm{T}}(N)R_n^{-1}r_n^b(N). \end{aligned} \tag{7.43}$$

Matricea $R_n(N)$ este semipozitiv definită, pentru că

$$\alpha^{\mathrm{T}} R_n(N)\alpha = \alpha^{\mathrm{T}} \sum_{t=n}^{N} \varphi_n(t)\varphi_n^{\mathrm{T}}(t)\alpha = \sum_{t=n}^{N} (\alpha^{\mathrm{T}}\varphi_n(t))^2 \geq 0, \tag{7.44}$$

iar datorită nesingularităţii rezultă că este pozitiv definită. Ca urmare, criteriile (7.42) şi (7.43) sunt minimizate de (7.38), respectiv (7.40), şi au valorile minime (7.39), respectiv (7.41).

□

Scanned with OKEN Scanner

Observaţii

1. Soluţia (7.38) poate fi privită şi ca o soluţie aproximativă a problemei de modelare AR cu criteriul de performanţă statistic (1.6) în care $E\varphi(t)\varphi^{\mathrm{T}}(t)$ se estimează prin $\frac{1}{N}\sum_{t=n}^{N}\varphi(t)\varphi^{\mathrm{T}}(t)$ iar $E\varphi(t)y(t)$ se estimează prin $\frac{1}{N}\sum_{t=n}^{N}\varphi(t)y(t)$.

2. Matricea $R_n(N)$ nu mai este matrice Toeplitz, dar are o structură apropiată de cea a unei matrice Toeplitz. Făcând ipoteza

$$y(t) = 0 \quad \text{pentru} \quad t \leq 0, \tag{7.45}$$

structura matricei R permite o rezolvare rapidă a problemei de modelare. Această ipoteză este cunoscută sub numele de fereastră anterioară (în [65], prewindow). În condiţiile ipotezei (7.45), se poate înlocui limitea inferioară de sumare de la $t = n$ la $t = 0$ în toate sumele care apar în relaţiile (7.32)–(7.38).

Structura vectorului observaţiilor oferă următoarele posibilităţi de recurenţă în ordin:

$$\varphi_{n+1}(t) = \begin{bmatrix} -y(t-1) \\ -y(t-2) \\ \cdot \\ \cdot \\ \cdot \\ -y(t-n-1) \end{bmatrix} = \begin{bmatrix} \varphi_n(t) \\ -y(t-n-1) \end{bmatrix} = \tag{7.46}$$

$$= \begin{bmatrix} -y(t-1) \\ \varphi_n(t-1) \end{bmatrix}. \tag{7.47}$$

Înlocuind în (7.34) relaţiile (7.46) şi (7.47), rezultă următoarele partiţionări:

$$R_{n+1}(N) = \begin{bmatrix} \sum_{t=n}^{N}\varphi_n(t)\varphi_n^{\mathrm{T}}(t) & -\sum_{t=n}^{N}\varphi_n(t)y(t-n-1) \\ -\sum_{t=n}^{N}\varphi_n^{\mathrm{T}}(t)y(t-n-1) & \sum_{t=n}^{N}y^2(t-n-1) \end{bmatrix} =$$

$$= \begin{bmatrix} R_n(N) & -r_n^b(N) \\ -r_n^{b\mathrm{T}}(N) & \rho_0(N-n-1) \end{bmatrix} = \tag{7.48}$$

$$= \begin{bmatrix} \sum_{t=n}^{N}y^2(t-1) & -\sum_{t=n}^{N}\varphi_n(t-1)y(t-1) \\ -\sum_{t=n}^{N}\varphi_n(t-1)y(t-1) & \sum_{t=n}^{N}\varphi_n(t-1)\varphi_n^{\mathrm{T}}(t-1) \end{bmatrix} =$$

$$= \begin{bmatrix} \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ -r_n(N-1) & R_n(N-1) \end{bmatrix}. \tag{7.49}$$

Utilizând în (7.35) relaţia (7.46), rezultă

$$r_{n+1}(N) = \begin{bmatrix} \sum_{t=n}^{N}\varphi_n(t)y(t) \\ -\sum_{t=n}^{N}y(t-n-1)y(t) \end{bmatrix} = \begin{bmatrix} r_n(N) \\ -\rho_{n+1}(N) \end{bmatrix}, \tag{7.50}$$

în care am notat

$$\rho_n(N) = \sum_{t=n}^{N} y(t)y(t-n). \tag{7.51}$$

Înlocuind în (7.36) relaţia (7.47), rezultă

$$r_{n+1}^b(N) = \begin{bmatrix} -\sum_{t=n}^{N} y(t-1)y(t-n-2) \\ -\sum_{t=n}^{N} \varphi_n(t-1)y(t-n-2) \end{bmatrix} = \begin{bmatrix} -\rho_{n+1}(N-1) \\ r_n^b(N-1) \end{bmatrix}. \tag{7.52}$$

Utilizând relaţia (7.49), predictorul optimal înainte şi "dispersia" optimală înainte vor rezulta ca soluţii ale sistemului extins

$$R_{n+1}(N)\begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} = \begin{bmatrix} \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ -r_n(N-1) & R_n(N-1) \end{bmatrix}\begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} = \begin{bmatrix} \sigma_n(N-1) \\ 0_n \end{bmatrix}, \tag{7.53}$$

iar sistemul extins pentru problema predicţiei înapoi se obţine utilizând partiţia (7.48) şi relaţiile (7.40), (7.41)

$$R_{n+1}(N)\begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} = \begin{bmatrix} R_n(N) & -r_n^b(N) \\ -r_n^{b\mathrm{T}}(N) & \rho_0(N-n-1) \end{bmatrix}\begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} = \begin{bmatrix} 0_n \\ \sigma_n^b(N) \end{bmatrix}. \tag{7.54}$$

Având în vedere asemănarea sistemelor de ecuaţii (7.53), (7.54) cu sistemele (1.19),(7.17), se pot obţine relaţii recursive similare celor din ALD, date de următoarea lemă.

Lema 7.2.2. *Soluţiile de ordin n şi $n+1$ ale sistemului de ecuaţii (7.53), (7.54) sunt legate prin relaţiile*

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \frac{\alpha_n(N)}{\sigma_n^b(N)}\begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}, \tag{7.55}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N-1) \end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n(N-1)}\begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix}, \tag{7.56}$$

$$\sigma_{n+1}(N) = \sigma_n(N) - \frac{\alpha_n^2(N)}{\sigma_n^b(N)}, \tag{7.57}$$

$$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \frac{\alpha_n^2(N-1)}{\sigma_n(N-1)}, \tag{7.58}$$

în care s-a notat

$$\alpha_n(N) = \rho_{n+1}(N) - r_n^{\mathrm{T}}(N)\theta_n^b(N). \tag{7.59}$$

Demonstraţie. Ţinând seama de (7.49),(7.50) şi (7.54)

$$R_{n+2}(N)\begin{bmatrix} 0 \\ \hline \theta_n^b(N-1) \\ 1 \end{bmatrix} = \left[\begin{array}{c|cc} \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) & \rho_{n+1}(N-1) \\ -r_n(N-1) & & \\ \rho_{n+1}(N-1) & R_{n+1}(N-1) & \end{array}\right]\left[\begin{array}{c} 0 \\ \hline \theta_n^b(N-1) \\ 1 \end{array}\right]$$

$$= \begin{bmatrix} \rho_{n+1}(N-1) - r_n^{\mathrm{T}}(N-1)\theta_n^b(N-1) \\ 0_n \\ \sigma_n^b(N-1) \end{bmatrix} = \begin{bmatrix} \alpha_n(N-1) \\ 0_n \\ \sigma_n^b(N-1) \end{bmatrix} \tag{7.60}$$

Scanned with OKEN Scanner

iar din relaţiile (7.48),(7.52) şi (7.53) rezultă

$$R_{n+2}(N)\begin{bmatrix}1\\ \theta_n(N-1)\\ 0\end{bmatrix} = \left[\begin{array}{c|c} R_{n+1}(N) & \begin{matrix}\rho_{n+1}(N-1)\\ -r_n^{bT}(N-1)\end{matrix} \\ \hline \begin{matrix}\rho_{n+1}(N-1) & -r_n^{bT}(N-1)\end{matrix} & \rho_0(N-n-2)\end{array}\right]\begin{bmatrix}1\\ \theta_n(N-1)\\ 0\end{bmatrix}$$

$$= \begin{bmatrix}\sigma_n(N-1)\\ 0_n\\ \rho_{n+1}(N-1) - r_n^{bT}(N-1)\theta_n(N-1)\end{bmatrix} = \begin{bmatrix}\sigma_n(N-1)\\ 0_n\\ \alpha_n(N-1)\end{bmatrix} \quad (7.61)$$

Combinând relaţiile (7.60) şi (7.61), se obţine

$$R_{n+2}(N)\left\{\begin{bmatrix}1\\ \theta_n(N-1)\\ 0\end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n^b(N-1)}\begin{bmatrix}0\\ \theta_n^b(N-1)\\ 1\end{bmatrix}\right\} = \begin{bmatrix}\sigma_n(N-1) - \frac{\alpha_n^2(N-1)}{\sigma_n^b(N-1)}\\ 0_n\\ 0\end{bmatrix}, \quad (7.62)$$

relaţie având aceeaşi formă cu (7.53) scrisă pentru $(n+1) \leftarrow n$

$$R_{n+2}(N)\begin{bmatrix}1\\ \theta_{n+1}(N-1)\end{bmatrix} = \begin{bmatrix}\sigma_{n+1}(N-1)\\ 0_{n+1}\end{bmatrix}. \quad (7.63)$$

Datorită nesingularităţii matricei $R_{n+2}(N)$, soluţia sistemului (7.63) va fi unică şi prin identificarea între (7.62) şi (7.63), rezultă (7.55) şi (7.57).

Sistemul extins (7.54) se scrie pentru $(n+1) \leftarrow n$

$$R_{n+2}(N)\begin{bmatrix}\theta_{n+1}^b(N)\\ 1\end{bmatrix} = \begin{bmatrix}0_{n+1}\\ \sigma_{n+1}^b(N)\end{bmatrix}. \quad (7.64)$$

O relaţie asemănătoare rezultă din următoarea combinare a relaţiilor (7.60) şi (7.61)

$$R_{n+2}(N)\left\{\begin{bmatrix}0\\ \theta_n^b(N-1)\\ 1\end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n(N-1)}\begin{bmatrix}1\\ \theta_n(N-1)\\ 0\end{bmatrix}\right\} = \begin{bmatrix}0\\ 0_n\\ \sigma_n^b(N-1) - \frac{\alpha_n^2(N-1)}{\sigma_n(N-1)}\end{bmatrix}. \quad (7.65)$$

Soluţia sistemului (7.64) fiind unică, prin identificare între relaţiile (7.64) şi (7.65) rezultă (7.56) şi (7.58).

□

Trebuie remarcat că Lema 7.2.2. nu furnizează direct un algoritm pentru calculul recursiv în ordin al predictorilor, datorită schimbării timpului curent din $N-1$ în N în relaţiile (7.56) şi (7.58), însă rezolvarea acestei probleme va fi obiectul secţiunii următoare. Relaţiile recursive (7.55)–(7.59) vor fi utilizate pentru deducerea de ecuaţii recursive ale predicţiilor.

Pentru a defini erorile de predicţie înainte, vom ţine cont de existenţa a două posibilităţi de utilizare a şirului de predictori optimali, indexat după t:

- predictorul $\theta_n(N-1)$, determinat de informaţia conţinută în setul $\{y(t)\}_{t=1}^{N-1}$, este utilizat pentru predicţia valorii $y(N)$, rezultând eroarea de predicţie înainte *apriori*

$$e_n^f(N) = y(N) - \varphi_n^T(N)\theta_n(N-1), \quad (7.66)$$

– predictorul $\theta_n(N)$, determinat din setul $\{y(t)\}_{t=1}^{N}$, este utilizat pentru "predicţia" valorii $y(N)$ (de exemplu în scopul filtrării), rezultând eroarea de predicţie înainte *aposteriori*

$$\varepsilon_n^f(N) = y(N) - \varphi_n^{\mathrm{T}}(N)\theta_n(N). \tag{7.67}$$

În mod similar se poate defini eroarea de predicţie înapoi *apriori* (depinzând de $\theta_n^b(N-1)$)

$$e_n^b(N) = y(N-n-1) - \varphi_n^{\mathrm{T}}(N)\theta_n^b(N-1) \tag{7.68}$$

şi eroarea de predicţie înapoi *aposteriori* (depinzând de $\theta_n^b(N)$)

$$\varepsilon_n^b(N) = y(N-n-1) - \varphi_n^{\mathrm{T}}(N)\theta_n^b(N). \tag{7.69}$$

Lema 7.2.3. *Erorile de predicţie (7.66)-(7.69) sunt legate prin următoarele relaţii recursive în ordin:*

$$\begin{aligned}
e_{n+1}^f(N) &= e_n^f(N) + K_{n+1}^f(N-1)e_n^b(N), && (7.70)\\
e_{n+1}^b(N) &= e_n^b(N-1) + K_{n+1}^b(N-2)e_n^f(N-1), && (7.71)\\
\varepsilon_{n+1}^f(N) &= \varepsilon_n^f(N) + K_{n+1}^f(N)\varepsilon_n^b(N), && (7.72)\\
\varepsilon_{n+1}^b(N) &= e_n^b(N-1) + K_{n+1}^b(N-1)\varepsilon_n^f(N-1), && (7.73)
\end{aligned}$$

în care s-au folosit notaţiile: coeficientul de reflexie înainte

$$K_{n+1}^f(N) = -\frac{\alpha_r(N)}{\sigma_r^b(N)} \tag{7.74}$$

şi coeficientul de reflexie înapoi:

$$K_{n+1}^b(N) = -\frac{\alpha_n(N)}{\sigma_n(N)}. \tag{7.75}$$

Demonstraţie. Utilizând (7.55) şi (7.46) în definiţia (7.66)

$$\begin{aligned}
e_{n+1}^f(N) &= y(N) - \varphi_{n+1}^{\mathrm{T}}(N)\theta_{n+1}(N-1) = \\
&= y(N) - \begin{bmatrix} \varphi_n^{\mathrm{T}}(N) & -y(N-n-1) \end{bmatrix} \left\{ \begin{bmatrix} \theta_n(N-1) \\ 0 \end{bmatrix} + K_{n+1}^f(N-1) \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix} \right\} \\
&= e_n^f(N) + K_{n+1}^f(N-1)e_n^b(N).
\end{aligned}$$

Utilizând (7.55) şi (7.46) în definiţia (7.67), rezultă

$$\begin{aligned}
\varepsilon_{n+1}^f(N) &= y(N) - \varphi_{n+1}^{\mathrm{T}}(N)\theta_{n+1}(N) = \\
&= y(N) - \begin{bmatrix} \varphi_n^{\mathrm{T}}(N) & -y(N-n-1) \end{bmatrix} \left\{ \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + K_{n+1}^f(N) \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \right\} = \\
&= \varepsilon_n^f(N) + K_{n+1}^f(N)\varepsilon_n^b(N).
\end{aligned}$$

Scanned with OKEN Scanner

Recursia erorilor de predicţie înapoi rezultă din recursia (7.56) şi relaţia (7.47):

$$\begin{aligned} e^b_{n+1}(N) &= y(N-n-2) - \varphi^{\mathrm{T}}_{n+1}(N)\theta^b_{n+1}(N-1) = \\ &= y(N-n-2) - \begin{bmatrix} -y(N-1) & \varphi^{\mathrm{T}}_r(N-1) \end{bmatrix} \left\{ \begin{bmatrix} 0 \\ \theta^b_n(N-2) \end{bmatrix} + \right. \\ &+ \left. K^b_{n+1}(N-2) \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix} \right\} = \varepsilon^b_n(N-1) + K^b_{n+1}(N-2)e^f_n(N-1), \end{aligned}$$

$$\begin{aligned} \varepsilon^b_{n+1}(N) &= y(N-n-2) - \varphi^{\mathrm{T}}_{n+1}(N)\theta^b_{n+1}(N) = \\ &= y(N-n-2) - \begin{bmatrix} -y(N-1) & \varphi^{\mathrm{T}}_r(N-1) \end{bmatrix} \left\{ \begin{bmatrix} 0 \\ \theta^b_n(N-1) \end{bmatrix} + \right. \\ &+ \left. K^b_{n+1}(N-1) \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \right\} = \varepsilon^b_n(N-1) + K^b_{n+1}(N-1)\varepsilon^f_n(N-1). \end{aligned}$$

□

Pentru obţinerea unui sistem complet de ecuaţii, formând un algoritm al erorilor de predicţie, este necesară o relaţie recursivă pentru calculul mărimii $\alpha_n(N)$. Următoarea lemă stabileşte această ecuaţie.

Lema 7.2.4. *Între valorile la momentele N şi $N-1$ ale mărimii α_n există următoarea relaţie*

$$\alpha_n(N) = \alpha_n(N-1) + \varepsilon^f_n(N)\varepsilon^b_n(N). \tag{7.76}$$

Demonstraţie. Relaţia de definiţie (7.59) se poate rescrie

$$\alpha_n(N) = \rho_{n+1}(N) - r^{\mathrm{T}}_n(N)R^{-1}_n(N)r^b_n(N). \tag{7.77}$$

Recursia în timp a mărimilor care intervin în această definiţie poate fi descrisă prin factorizarea

$$\begin{bmatrix} 1 & -y(N-n-1) & \varphi^{\mathrm{T}}_n(N) \\ -y(N) & \rho_{n+1}(N) & -r^{\mathrm{T}}_n(N) \\ \varphi_n(N) & -r^b_n(N) & R_n(N) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -y(N) & 1 & 0 \\ \varphi_n(N) & 0 & I \end{bmatrix} \cdot$$
$$\cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho_{n+1}(N-1) & -r^{\mathrm{T}}_n(N-1) \\ 0 & -r^b_n(N-1) & R_n(N-1) \end{bmatrix} \begin{bmatrix} 1 & -y(N-n-1) & \varphi^{\mathrm{T}}_n(N) \\ 0 & 1 & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{7.78}$$

iar înmulţind la stânga cu inversa primei matrice din membrul drept se obţine

$$\begin{bmatrix} 1 & 0 & 0 \\ y(N) & 1 & 0 \\ \varphi_n(N) & 0 & I \end{bmatrix} \begin{bmatrix} 1 & -y(N-n-1) & \varphi^{\mathrm{T}}_n(N) \\ -y(N) & \rho_{n+1}(N) & -r^{\mathrm{T}}_n(N) \\ \varphi_n(N) & -r^b_n(N) & R_n(N) \end{bmatrix} =$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho_{n+1}(N-1) & -r^{\mathrm{T}}_n(N-1) \\ 0 & -r^b_n(N-1) & R_n(N-1) \end{bmatrix} \begin{bmatrix} 1 & -y(N-n-1) & \varphi^{\mathrm{T}}_n(N) \\ 0 & 1 & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{7.79}$$

Scanned with OKEN Scanner

Înmulţind (7.79) la dreapta cu $\left[\begin{array}{ccc} 0 & 1 & \theta_n^{b\mathrm{T}}(N)\end{array}\right]^{\mathrm{T}}$ şi la stânga cu $\left[\begin{array}{ccc} 0 & 1 & \theta_n^{\mathrm{T}}(N-1)\end{array}\right]$ rezultă

$$\left[\begin{array}{ccc} e_n^f(N) & 1 & \theta_n^{\mathrm{T}}(N-1)\end{array}\right]\left[\begin{array}{c} -\varepsilon_n^b(N) \\ \alpha_n(N) \\ 0\end{array}\right]=\left[\begin{array}{ccc} 0 & \alpha_n(N-1) & 0\end{array}\right]\left[\begin{array}{c} -\varepsilon_n^b(N) \\ 1 \\ \theta_n^b(N)\end{array}\right], \tag{7.80}$$

deci

$$\alpha_n(N)=\alpha_n(N-1)+e_n^f(N)\varepsilon_n^b(N).$$

Grupând relaţiile (7.57),(7.58),(7.70)–(7.75) şi (7.76) rezultă algoritmul recursiv în timp şi ordin al erorilor de predicţie din Tabelul 7.3.

Structura acestui algoritm cuprinde două filtre latice, unul pentru calculul erorilor apriori, celălalt pentru calculul erorilor aposteriori. Filtrele latice păstrează structura filtrului din Fig. 7.1.*b*, dar coeficienţii de reflexie din cadrul aceluiaşi etaj sunt diferiţi: $K_{n+1}^f \neq K_{n+1}^b$. Relaţiile (7.57) şi (7.58) de calcul a "dispersiilor" de eşantion a erorilor de predicţie formează o a treia structură latice:

$$\sigma_{n+1}(N) = \sigma_n(N)-K_{n+1}^{f2}(N)\sigma_n^b(N), \tag{7.81}$$

$$\sigma_{n+1}^b(N) = \sigma_n^b(N-1)-K_{n+1}^{b2}(N-1)\sigma_n(N-1), \tag{7.82}$$

dar ar necesita astfel 4 înmulţiri şi două scăderi faţă de 3 înmulţiri şi o scădere în ecuaţiile (6) şi (7) din Tabelul 7.3.

Iniţializarea recursiilor pentru ordinul fictiv 0 asigură pentru ordinul $n=1$ *exact* valorile de definiţie ale tuturor mărimilor din algoritm, de exemplu din

$$\sigma_0(N)=\rho_0(N);\quad \sigma_0^b(N)=\rho_0(N-1);\quad \alpha_0(N)=\rho_1(N),$$

rezultă:

$$K_1^f(N) = -\frac{\rho_1(N)}{\rho_0(N-1)}=\frac{r_1(N)}{R_1(N)}=\theta_1(N), \tag{7.83}$$

$$K_1^b(N) = -\frac{\rho_1(N)}{\rho_0(N)}=\frac{r_1^b(N+1)}{R_1(N+1)}=\theta_1^b(N+1), \tag{7.84}$$

$$\sigma_1^b(N) = \rho_0(N-2)-\frac{\rho_1^2(N-1)}{\rho_0(N-1)}=\rho_0(N-2)-r_1^b(N)\theta_1^b(N), \tag{7.85}$$

$$e_1^f(N) = y(N)+\theta_1^f(N-1)y(N-1). \tag{7.86}$$

Observaţii

1. Din (7.53),(7.54) se observă că atunci când $\sigma_n(N)$ sau $\sigma_n^b(N)$ sunt nule, matricea R_{n+1} este singulară. Această problemă este tratată în detaliu la sfârşitul secţiunii următoare, unde se justifică şi modul de tratare folosit în algoritmul 7.2.1., pentru situaţiile $\sigma_n=0$ sau $\sigma_n^b=0$, care conduce la un algoritm CMMP *exact*.

2. În multe aplicaţii este nevoie numai de erorile de predicţie, nu şi de valorile predicţiilor. În cazul în care este necesară, de exemplu, predicţia înainte de ordin n, aceasta poate fi calculată cu oricare din relaţiile:

$$\tilde{y}_n(N) = y(N)-\varepsilon_n^f(N), \tag{7.87}$$

$$\tilde{y}_n(N) = y(N)-e_n^f(N). \tag{7.88}$$

Scanned with OKEN Scanner

Algoritmul 7.2.1.

ALGORITM LATICE AL ERORILOR DE PREDICTIE OPTIMAL		*Număr operaţii/ ecuaţie* ⋆ / +	*Număr operaţii/ pas alg.*
1. *Iniţializare la momentul* $(N = 0)$			
Pentru $n = 0, \ldots, n_{max} - 1$			
$\alpha_n(0) = 0$			
$\sigma_0(0) = \sigma_0^b(0) = 0$			
2. *Recursii în timp*			$\mathcal{O}(n_{max}^2)$
Pentru $N = 1, 2, \ldots, N_{max}$			
2.1 *Iniţializare recursii în ordin*			
$e_0^f(N) = \varepsilon_0^f(N) = y(N)$			
$e_0^b(N) = \varepsilon_0^b(N) = y(N-1)$			
$\sigma_0(N) = \sigma_0(N-1) + \varepsilon_0^{f2}(N)$	(1)	1 0 1	
$\sigma_0^b(N) = \sigma_0(N-1)$	(2)		
2.2 *Recursii în ordin:*			$\mathcal{O}(n_{max})$
Pentru $n = 0, \ldots, n_{max} - 1$			
$\alpha_n(N) = \alpha_n(N-1) + e_n^f(N)\varepsilon_n^b(N)$	(3)	1 0 1	
$K_{n+1}^f(N) = -\alpha_n(N)/\sigma_n^b(N)$ $(\star)$	(4)	0 1 0	
$K_{n+1}^b(N) = -\alpha_n(N)/\sigma_n(N)$ $(\star)$	(5)	0 1 0	
$\sigma_{n+1}(N) = \sigma_n(N)(1 - K_{n+1}^f(N)K_{n+1}^b(N))$	(6)	2 0 1	
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1)(1 - K_{n+1}^f(N-1)K_{n+1}^b(N-1))$	(7)	1 0 0	
$e_{n+1}^f(N) = e_n^f(N) + K_{n+1}^f(N-1)e_n^b(N)$	(8)	1 0 1	
$e_{n+1}^b(N) = e_n^b(N-1) + K_{n+1}^b(N-2)e_n^f(N-1)$	(9)	1 0 1	
$\varepsilon_{n+1}^f(N) = \varepsilon_n^f(N) + K_{n+1}^f(N)\varepsilon_n^b(N)$	(10)	1 0 1	
$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) + K_{n+1}^b(N-1)\varepsilon_n^f(N-1)$	(11)	1 0 1	
$(\star)$: *Dacă* $\sigma_n^b(N) = 0$ *se face* $K_{n+1}^f(N) = 0$			
Dacă $\sigma_n(N) = 0$ *se face* $K_{n+1}^b(N) = 0$			
Total: $8n_{max}$ *înmulţiri;* $2n_{max}$ *împărţiri;*$6n_{max}$ *adunări*			

Tabelul 7.3. Algoritmul latice al erorilor de predicţie optimal.

Scanned with OKEN Scanner

Prima relaţie este de aşteptat să producă estimaţii mai bune, pentru că în estimaţia aposteriori este utilizat predictorul dedus dintr-o informaţie mai "bogată".

3. Algoritmul 7.2.1. este aproximativ de două ori mai costisitor decât Algoritmul 7.1.2., dar este de preferat în toate aplicaţiile în care sunt cerute estimaţii precise după un număr relativ mic de paşi.

7.3. Algoritmul latice al predictorilor

În această secţiune vor fi discutate alte soluţii ale Problemei 7.2.1. şi în plus se vor prezenta algoritmi pentru rezolvarea eficientă a următoarei probleme:

Problema 7.3.1. *Fiind dată seria* $\{y(t)\}$, *să se găsească predictorii înainte* $\{\theta_n(N)\}_{n=1}^{n_{max}}$ *care minimizează (7.32) şi predictorii înapoi* $\{\theta_n^b(N)\}_{n=1}^{n_{max}}$ *care minimizează (7.33), utilizând algoritmi bloc recursivi în timp.*

Obţinerea eficientă a tuturor predictorilor optimali de ordine între 1 şi n_{max} se poate realiza prin ecuaţiile recursive în ordin (7.55)–(7.56) în $\mathcal{O}(n_{max}^2)$ operaţii; în iterarea acestor recursii sunt necesari coeficienţii de reflexie care se calculează în $\mathcal{O}(n_{max})$ operaţii prin iteraţia în ordin din cadrul algoritmului latice al erorilor de predicţie. Metoda prezentată ar necesita $\mathcal{O}(n_{max}^2)$ operaţii la fiecare moment de eşantionare al seriei $y(N)$.

Există aplicaţii în care modelele AR sunt necesare periodic, dar la intervale mult mai mari de timp decât perioada de eşantionare (de culegere a datelor). Metodele care realizează astfel de prelucrări se numesc metode bloc recursive în timp. Să observăm mai întâi că schema următoare: algoritm latice + relaţiile (7.55),(7.56) nu furnizează o metodă bloc recursivă, pentru că $\theta_{n+1}^b(N)$ este calculat în relaţia (7.56) în funcţie de $\theta_n^b(N-1)$, care nu este disponibil decât dacă predictorii sunt evaluaţi la fiecare moment de eşantionare.

Metodele rapide bloc recursive se bazează pe propagarea la fiecare moment de eşantionare a informaţiei utile (de exemplu prin algoritmul latice sunt propagaţi coeficienţii de reflexie în $\mathcal{O}(n_{max})$ operaţii) şi numai la momentele în care se cer şi predictorii (modelele AR) se execută un algoritm recursiv în ordin necesitând $\mathcal{O}(n_{max}^2)$ operaţii. Există numeroase variante posibile de combinare a algoritmilor de tip latice cu recursii în ordin ale predictorilor; aceste variante vor fi uşor obţinute după ce, printr-o abordare sistematică, se vor deduce toate relaţiile recursive în timp şi în ordin.

7.3.1. Deducerea ecuaţiilor de recurenţă în timp

Pentru stabilirea ecuaţiilor de recurenţă în timp se introduc următoarele mărimi auxiliare:

- $K_n(N)$, numit vectorul amplificării Kalman apriori, şi definit ca soluţie a sistemului

$$R_n(N-1)K_n(N) = \varphi_n(N); \tag{7.89}$$

- $K_n^*(N)$, numit vectorul amplificării Kalman aposteriori, şi definit ca soluţie a sistemului

$$R_n(N)K_n^*(N) = \varphi_n(N); \tag{7.90}$$

Scanned with OKEN Scanner

- $\gamma_n^*(N)$, numit amplificare aposteriori, definit de

$$\gamma_n^*(N) = \varphi_n^{\mathrm{T}}(N) K_n(N); \tag{7.91}$$

- $\gamma_n(N)$, numit amplificare apriori, definit de

$$\gamma_n(N) = \varphi_n^{\mathrm{T}}(N) K_n(N); \tag{7.92}$$

- $\gamma_n^0(N)$, numit raport aposteriori/apriori, definit de

$$\gamma_n^0(N) = 1 - \gamma_n^*(N). \tag{7.93}$$

Observaţie. Pentru a justifica terminologia pe care o vom utiliza mai târziu (de exemplu $K_n(N)$ este considerat un predictor), dăm următoarea interpretare mărimilor definite mai sus. Considerăm problema de predicţie a seriei

$$x(t) = \begin{cases} 0 & t = 0, \ldots, N-1 \\ 1 & t = N \end{cases}$$

printr-o combinaţie liniară a elementelor vectorului $\varphi_n(t)$

$$\hat{x}(t) = K^{\mathrm{T}} \varphi_n(t).$$

Vom evalua precizia predicţiei prin criteriul

$$\sum_{t=0}^{N} (x(t) - \hat{x}(t))^2 = \sum_{t=0}^{N} (x(t) - K^{\mathrm{T}} \varphi_n(t))^2.$$

Se poate demonstra uşor (într-un mod similar demonstraţiei Lemei 7.2.1.) că predictorul care minimizează criteriul de mai sus este

$$K_n(N) = R_n^{-1}(N) \varphi_n(N),$$

iar dispersia de eşantion a erorilor de predicţie optimale este

$$\gamma_n^0(N) = 1 - \varphi_n^{\mathrm{T}}(N) K_n(N).$$

Relaţiile care descriu recurenţa în timp a predictorilor şi a dispersiilor erorilor precum şi alte relaţii legate de recurenţa în timp sunt prezentate în Tabelul 7.4. În continuare va fi abordată problema deducerii acestor relaţii (exceptând relaţia (7.98) care a fost deja demonstrată în Lema 7.2.4.).

Recurenţa în timp a matricei $R_n(N)$ poate fi descrisă prin factorizarea

$$\begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N+1) \\ \varphi_n(N+1) & R_n(N+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \varphi_n(N+1) & I_n \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & R_n(N) \end{bmatrix} \begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N+1) \\ 0 & I_n \end{bmatrix}, \tag{7.103}$$

de unde rezultă, prin înmulţire la stânga cu inversa primei matrice din membrul drept, relaţia

$$\begin{bmatrix} 1 & 0 \\ -\varphi_n(N+1) & I_n \end{bmatrix} \begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N+1) \\ \varphi_n(N+1) & R_n(N+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & R_n(N) \end{bmatrix} \begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N+1) \\ 0 & I_n \end{bmatrix}, \tag{7.104}$$

Scanned with OKEN Scanner

$$\theta_n(N) = \theta_n(N-1) + K_n(N)\varepsilon_n^f(N) \quad (7.94)$$

$$\theta_n^b(N) = \theta_n^b(N-1) + K_n(N)\varepsilon_n^b(N) \quad (7.95)$$

$$\sigma_n(N) = \sigma_n(N-1) + e_n^f(N)\varepsilon_n^f(N) \quad (7.96)$$

$$\sigma_n^b(N) = \sigma_n^b(N-1) + e_n^b(N)\varepsilon_n^b(N) \quad (7.97)$$

$$\alpha_n(N) = \alpha_n(N-1) + e_n^f(N)\varepsilon_n^b(N) \quad (7.98)$$

$$K_n^*(N) = K_n(N)\gamma_n^0(N) \quad (7.99)$$

$$\varepsilon_n^f(N) = e_n^f(N)\gamma_n^0(N) \quad (7.100)$$

$$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N) \quad (7.101)$$

$$\gamma_n^0(N) = 1 - \gamma_n^*(N) = \frac{1}{1+\gamma_n(N)} \quad (7.102)$$

Tabelul 7.4. Relaţiile de bază pentru recurenţa în timp.

Ecuaţia de bază	Partiţionarea matricei R_{n+1}	Înmulţire la stânga	Înmulţire la dreapta	Ecuaţia rezultată
(7.104) $(n+1 \leftarrow n)$	(7.49),(7.47)	$\begin{bmatrix} 0 & 0 & I \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N) \end{bmatrix}^{\mathrm{T}}$	(7.94)
(7.104) $(n+1 \leftarrow n)$	(7.49),(7.47)	$\begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N) \end{bmatrix}^{\mathrm{T}}$	(7.96)
(7.104) $(n+1 \leftarrow n)$	(7.48),(7.46)	$\begin{bmatrix} 0 & I & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & \theta_n^{b\mathrm{T}}(N+1) & 1 \end{bmatrix}^{\mathrm{T}}$	(7.95)
(7.104) $(n+1 \leftarrow n)$	(7.48),(7.46)	$\begin{bmatrix} 0 & \theta_n^{b\mathrm{T}}(N) & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & \theta_n^{b\mathrm{T}}(N+1) & 1 \end{bmatrix}^{\mathrm{T}}$	(7.97)
(7.104) $(n+1 \leftarrow n)$	(7.49),(7.47)	$\begin{bmatrix} 0 & 0 & I \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -K_n^{*\mathrm{T}}(N) \end{bmatrix}^{\mathrm{T}}$	(7.99)
(7.104) $(n+1 \leftarrow n)$	(7.49),(7.47)	$\begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -K_n^{*\mathrm{T}}(N) \end{bmatrix}^{\mathrm{T}}$	(7.100)
(7.104) $(n+1 \leftarrow n)$	(7.48),(7.46)	$\begin{bmatrix} 0 & \theta_n^{b\mathrm{T}}(N) & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -K_n^{*\mathrm{T}}(N+1) & 0 \end{bmatrix}^{\mathrm{T}}$	(7.101)

Tabelul 7.5. Operaţii necesare pentru deducerea ecuaţiilor de recurenţă în timp.

Scanned with OKEN Scanner

care constituie o ecuaţie de bază, din care vor fi deduse, prin înmulţiri la dreapta sau la stânga cu matrice convenabil alese, toate ecuaţiile din Tabelul 7.4. În Tabelul 7.5. sunt prezentate sintetic operaţiile necesare, care vor fi detaliate în cele ce urmează.

Prima linie a Tabelului 7.5. indică efectuarea următoarelor operaţii

$$\left[\begin{array}{ccc} 0 & 0 & I_n \end{array}\right] \left[\begin{array}{ccc} 1 & 0 & 0 \\ y(N) & 1 & 0 \\ -\varphi_n(N) & 0 & I_n \end{array}\right] \left[\begin{array}{c|c} 1 & -y(N)\ \varphi_n^{\mathrm{T}}(N) \\ \hline -y(N) & \\ \varphi_n(N) & R_{n+1}(N+1) \end{array}\right] \left[\begin{array}{c} 0 \\ \hline 1 \\ \theta_n(N) \end{array}\right] = \tag{7.105}$$

$$= \left[\begin{array}{ccc} 0 & 0 & I_n \end{array}\right] \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ 0 & -r_n(N-1) & R_n(N-1) \end{array}\right] \left[\begin{array}{ccc} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ 0 & 1 & 0 \\ 0 & 0 & I_n \end{array}\right] \left[\begin{array}{c} 0 \\ 1 \\ \theta_n(N) \end{array}\right], \tag{7.106}$$

rezultând

$$\left[\begin{array}{ccc} -\varphi_n(N) & 0 & I_n \end{array}\right] \left[\begin{array}{c} -\varepsilon_n^f(N) \\ \sigma_n(N) \\ 0_n \end{array}\right] = \left[\begin{array}{ccc} 0 & -r_n(N-1) & R_n(N-1) \end{array}\right] \left[\begin{array}{c} -\varepsilon_n^f(N) \\ 1 \\ \theta_N(n) \end{array}\right], \tag{7.107}$$

sau

$$\varphi_n(N)\varepsilon_n^f(N) = -r_n(N-1) + R_n(N-1)\theta_n(N), \tag{7.108}$$

iar prin utilizarea relaţiilor de definiţie (7.38) şi (7.89) rezultă

$$R_n(N-1)K_n(N)\varepsilon_n^f(N) = -R_n(N-1)\theta_n(N-1) + R_n(N-1)\theta_n(N). \tag{7.109}$$

Una din soluţiile sistemului (7.109) (dacă $R_n(N-1)$ este nesingulară, atunci chiar unica soluţie) este dată de (7.94).

În a doua linie a Tabelului 7.5. sunt indicate următoarele operaţii

$$\left[\begin{array}{ccc} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{array}\right] \left[\begin{array}{ccc} 1 & 0 & 0 \\ y(N) & 1 & 0 \\ -\varphi_n(N) & 0 & I \end{array}\right] \left[\begin{array}{c|c} 1 & -y(N)\varphi_n^{\mathrm{T}}(N) \\ \hline -y(N) & \\ \varphi_n(N) & R_{n+1}(N+1) \end{array}\right] \left[\begin{array}{c} 0 \\ \hline 1 \\ \theta_n(N) \end{array}\right] = \tag{7.110}$$

$$= \left[\begin{array}{ccc} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{array}\right] \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ 0 & -r_n(N-1) & R_n(N-1) \end{array}\right] \left[\begin{array}{ccc} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ 0 & 1 & 0 \\ 0 & 0 & I_n \end{array}\right] \left[\begin{array}{c} 0 \\ 1 \\ \theta_n(N) \end{array}\right]$$

şi rezultă

$$\left[\begin{array}{ccc} e_n^f(N) & 1 & \theta_n^{\mathrm{T}}(N-1) \end{array}\right] \left[\begin{array}{c} -\varepsilon_n^f(N) \\ \sigma_n(N) \\ 0 \end{array}\right] = \left[\begin{array}{ccc} 0 & \sigma_n(N-1) & 0 \end{array}\right] \left[\begin{array}{c} -\varepsilon_n^f(N) \\ 1 \\ \theta_n(N) \end{array}\right], \tag{7.111}$$

sau

$$-e_n^f(N)\varepsilon_n^f(N) + \sigma_n(N) = \sigma_n(N-1),$$

această relaţie fiind echivalentă cu (7.96).

Calculele implicate în obţinerea relaţiilor (7.95) şi (7.97) nu le vom mai detalia, fiind foarte asemănătoare celor prezentate deja pentru obţinerea relaţiilor (7.94) şi (7.95).

Scanned with OKEN Scanner

În linia a 5-a a Tabelului 7.5. sunt indicate următoarele operaţii

$$\begin{bmatrix} 0 & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ y(N) & 1 & 0 \\ -\varphi_n(N) & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ -y(N) & \rho_0(N) & -r_n^{\mathrm{T}}(N) \\ \varphi_n(N) & -r_n(N) & R_n(N) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N) \end{bmatrix} = \tag{7.112}$$

$$= \begin{bmatrix} 0 & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ 0 & -r_n(N-1) & R_n(N-1) \end{bmatrix} \begin{bmatrix} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ 0 & 1 & 0 \\ 0 & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N) \end{bmatrix} \tag{7.113}$$

Notând că

$$\begin{aligned} -y(N) + r_n^{\mathrm{T}}(N) K_n^*(N) &= -y(N) + \theta_n^{\mathrm{T}}(N) R_n(N) K_n^*(N) = \\ &= -y(N) + \theta_n^{\mathrm{T}}(N) \varphi_n(N) = -\varepsilon_n^f(N), \end{aligned} \tag{7.114}$$

relaţia (7.112) se transformă în

$$\begin{bmatrix} -\varphi_n(N) & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 - \gamma_n^*(N) \\ -\varepsilon_n^f(N) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -r_n(N-1) & R_n(N-1) \end{bmatrix} \begin{bmatrix} 1 - \gamma_n^*(N) \\ 0 \\ -K_n^*(N) \end{bmatrix}, \tag{7.115}$$

sau

$$-\varphi_n(N)(1 - \gamma_n^*(N)) = -R_n(N-1) K_n^*(N), \tag{7.116}$$

şi utilizând definiţia (7.89), rezultă

$$R_n(N-1) K_n(N)(1 - \gamma_n^*(N)) = R_n(N-1) K_n^*(N), \tag{7.117}$$

care are ca soluţie (unică dacă $|R_n(N-1)| \neq 0$) relaţia (7.99).

Operaţiile indicate în linia a 6-a a Tabelului 7.5. sunt:

$$\begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ y(N) & 1 & 0 \\ -\varphi_n(N) & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ -y(N) & \rho_0(N) & -r_n^{\mathrm{T}}(N) \\ \varphi_n(N) & -r_n(N) & R_n(N) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N) \end{bmatrix} \tag{7.118}$$

$$= \begin{bmatrix} 0 & 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ 0 & -r_n(N-1) & R_n(N-1) \end{bmatrix} \begin{bmatrix} 1 & -y(N) & \varphi_n^{\mathrm{T}}(N) \\ 0 & 1 & 0 \\ 0 & 0 & I_n \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N) \end{bmatrix}$$

şi efectuând produsele rezultă

$$\begin{bmatrix} e_n^f(N) & 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix} \begin{bmatrix} 1 - \gamma_n^*(N) \\ -\varepsilon_n^f(N) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \sigma_n(N-1) & 0 \end{bmatrix} \begin{bmatrix} 1 - \gamma_n^*(N) \\ 0 \\ -K_n^*(N) \end{bmatrix}, \tag{7.119}$$

deci

$$e_n^f(N)(1 - \gamma_n^*(N)) - \varepsilon_n^f(N) = 0,$$

relaţie care este echivalentă cu (7.100). Operaţiile indicate în ultima linie a Tabelului 7.5. sunt foarte asemănătoare celor din linia a şasea, aşa încât nu vor mai fi detaliate.

Scanned with OKEN Scanner

În sfârşit, ultima relaţie, (7.102) din Tabelul 7.4. va rezulta imediat utilizând (7.99):

$$\begin{aligned}\gamma_n^*(N) &= \varphi_n^{\mathrm{T}}(N)K_n^*(N) = \varphi_n^{\mathrm{T}}(N)K_n(N)(1-\gamma_n^*(N)) = \\ &= \gamma_n(N)(1-\gamma_n^*(N)),\end{aligned} \tag{7.120}$$

deci

$$\gamma_n(N) = \frac{\gamma_n^*(N)}{1-\gamma_n^*(N)}, \tag{7.121}$$

$$\gamma_n^*(N) = \frac{\gamma_n(N)}{1+\gamma_n(N)}. \tag{7.122}$$

7.3.2. Deducerea ecuaţiilor recursive în ordin

În Lemele 7.2.2. şi 7.2.3. au fost deduse ecuaţii recursive în ordin pentru următoarele mărimi: predictori, dispersiile de eşantion, erorile de predicţie. În lema următoare vor fi stabilite ecuaţii recursive în ordin pentru vectorii Kalman şi amplificările apriori şi aposteriori.

Lema 7.3.1. *Au loc următoarele relaţii de recurenţă în ordin:*

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}, \tag{7.139}$$

$$K_{n+1}(N) = \begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} - \frac{e_n^b(N)}{\sigma_n^b(N-1)} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix}, \tag{7.140}$$

$$K_{n+1}^*(N) = \begin{bmatrix} 0 \\ K_n^*(N-1) \end{bmatrix} - \frac{\varepsilon_n^f(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix}, \tag{7.141}$$

$$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix}, \tag{7.142}$$

$$\gamma_{n+1}^*(N) = \gamma_n^*(N) + \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N)}, \tag{7.143}$$

$$\gamma_{n+1}(N) = \gamma_n(N) + \frac{e_n^{b2}(N)}{\sigma_n^b(N-1)}, \tag{7.144}$$

$$\gamma_{n+1}^*(N) = \gamma_n^*(N-1) + \frac{\varepsilon_n^{f2}(N-1)}{\sigma_n(N-1)}, \tag{7.145}$$

$$\gamma_{n+1}(N) = \gamma_n(N-1) + \frac{e_n^{f2}(N-1)}{\sigma_n(N-2)}. \tag{7.146}$$

Demonstraţie. Se va urma acelaşi procedeu ca cel utilizat pentru demonstrarea lemei 7.2.2. Definiţiile (7.89)-(7.92) scrise pentru predictorii K şi dispersiile $1-\gamma$ de ordin $n+1$ pot fi grupate în următoarele sisteme extinse:

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 1 \\ -K_{n+1}^*(N) \end{bmatrix} = \begin{bmatrix} 1-\gamma_{n+1}^*(N) \\ 0_{n+1} \end{bmatrix}, \tag{7.147}$$

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 1 \\ -K_{n+1}(N) \end{bmatrix} = \begin{bmatrix} 1-\gamma_{n+1}(N) \\ 0_{n+1} \end{bmatrix}. \tag{7.148}$$

Scanned with OKEN Scanner

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \frac{\alpha_n(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \tag{7.123}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N-1) \end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \tag{7.124}$$

$$\sigma_{n+1}(N) = \sigma_n(N) - \frac{\alpha_n^2(N)}{\sigma_n^b(N)} \tag{7.125}$$

$$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \frac{\alpha_n^2(N-1)}{\sigma_n(N-1)} \tag{7.126}$$

$$e_{n+1}^f(N) = e_n^f(N) - \frac{\alpha_n(N-1)}{\sigma_n^b(N-1)} e_n^b(N) \tag{7.127}$$

$$e_{n+1}^b(N) = e_n^b(N-1) - \frac{\alpha_n(N-2)}{\sigma_n(N-2)} e_n^f(N-1) \tag{7.128}$$

$$\varepsilon_{n+1}^f(N) = \varepsilon_n^f(N) - \frac{\alpha_n(N)}{\sigma_n^b(N)} \varepsilon_n^b(N) \tag{7.129}$$

$$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} \varepsilon_n^f(N-1) \tag{7.130}$$

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{e_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \tag{7.131}$$

$$K_{n+1}(N) = \begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} - \frac{e_n^b(N)}{\sigma_n^b(N-1)} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix} \tag{7.132}$$

$$K_{n+1}^*(N) = \begin{bmatrix} 0 \\ K_n^*(N-1) \end{bmatrix} - \frac{\varepsilon_n^f(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \tag{7.133}$$

$$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix} \tag{7.134}$$

$$\gamma_{n+1}^*(N) = \gamma_n^*(N) + \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N)} \tag{7.135}$$

$$\gamma_{n+1}(N) = \gamma_n(N) + \frac{e_n^{b2}(N)}{\sigma_n^b(N-1)} \tag{7.136}$$

$$\gamma_{n+1}^*(N) = \gamma_n^*(N-1) + \frac{\varepsilon_n^{f2}(N-1)}{\sigma_n(N-1)} \tag{7.137}$$

$$\gamma_{n+1}(N) = \gamma_n(N-1) + \frac{e_n^{f2}(N-1)}{\sigma_n(N-2)} \tag{7.138}$$

Tabelul 7.6. Relaţiile de bază pentru recurenţa în ordin.

Scanned with OKEN Scanner

Utilizând partiţia (obţinută cu (7.46) şi (7.48))

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} = \begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N) & -y(N-n-1) \\ \varphi_n(N) & R_n(N) & -r_n^b(N) \\ -y(N-n-1) & -r_n^{b\mathrm{T}}(N) & \rho_0(N-n-1) \end{bmatrix}, \tag{7.149}$$

se stabilesc imediat identităţile:

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 0 \\ \theta_n^b(N) \\ 1 \end{bmatrix} = \begin{bmatrix} -\varepsilon_n^b(N) \\ 0_n \\ \sigma_n^b(N) \end{bmatrix}, \tag{7.150}$$

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 1 \\ -K_n^*(N) \\ 0 \end{bmatrix} = \begin{bmatrix} 1-\gamma_n^*(N) \\ 0_n \\ -\varepsilon_n^b(N) \end{bmatrix}, \tag{7.151}$$

care pot fi combinate astfel

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \left\{ \begin{bmatrix} 1 \\ -K_n^*(N) \\ 0 \end{bmatrix} + \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} 0 \\ \theta_n^b(N) \\ 1 \end{bmatrix} \right\} = \begin{bmatrix} 1-\gamma_n^*(N) - \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N)} \\ 0_{n+1} \end{bmatrix} \tag{7.152}$$

şi comparând cu (7.147) rezultă (7.139) şi (7.143).

Utilizând partiţia (obţinută cu relaţiile (7.46) şi (7.48))

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} = \begin{bmatrix} 1 & \varphi_n^{\mathrm{T}}(N) & -y(N-n-1) \\ \varphi_n(N) & R_n(N-1) & -r_n^b(N-1) \\ -y(N-n-1) & -r_n^{b\mathrm{T}}(N-1) & \rho_0(N-n-2) \end{bmatrix}, \tag{7.153}$$

rezultă identităţile:

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \begin{bmatrix} 1 \\ -K_n(N) \\ 0 \end{bmatrix} = \begin{bmatrix} 1-\gamma_n(N) \\ 0_n \\ -e_n^b(N) \end{bmatrix}, \tag{7.154}$$

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \begin{bmatrix} 0 \\ \theta_n^b(N-1) \\ 1 \end{bmatrix} = \begin{bmatrix} -e_n^b(N) \\ 0_n \\ \sigma_n^b(N-1) \end{bmatrix}, \tag{7.155}$$

care pot fi combinate pentru a rezulta

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \left\{ \begin{bmatrix} 1 \\ -K_n(N) \\ 0 \end{bmatrix} + \frac{e_n^b(N)}{\sigma_n^b(N-1)} \begin{bmatrix} 0 \\ \theta_n^b(N-1) \\ 1 \end{bmatrix} \right\} =$$

$$= \begin{bmatrix} 1-\gamma_n(N) - \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N-1)} \\ 0_{n+1} \end{bmatrix}. \tag{7.156}$$

Comparând (7.156) cu (7.148), rezultă (7.140) şi (7.144).

Scanned with OKEN Scanner

Extindere a Algoritmului 7.1.2.

SECŢIUNEA OPŢIONALĂ PENTRU RECURSIA ÎN ORDIN A PREDICTORILOR	Număr operaţii/ ecuaţie ⋆	/	+
2.3 Dacă se doreşte evaluarea predictorilor la momentul N Atunci:			
2.3.1 Iniţializare recursii în ordin:			
$\theta_1(N) = -\alpha_0(N)/\sigma_0^b(N)$	1		
$\theta_1^b(N) = -\alpha_0(N-1)/\sigma_0(N-1)$	1		
$K_1^*(N) = -\varepsilon_0^b(N)/\sigma_0^b(N)$	1		
2.3.2 Recursii în ordin:			
Pentru $n = 1, \ldots, n_{max} - 1$			
$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \frac{\alpha_n(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$	n	1	n
$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} - (\varepsilon_n^b(N) - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} e_n^f(N)) \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix}$	$2n+1$	1	$2n+1$
$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$	n	1	n
Total: $\mathcal{O}(n_{max}^2)$ înmulţiri; $\mathcal{O}(n_{max})$ împărţiri; $\mathcal{O}(n_{max}^2)$ adunări			

Tabelul 7.7. Completarea algoritmului latice al erorilor de predicţie optimal cu o secţiune recursivă în ordin de calcul a predictorilor optimali.

Scanned with OKEN Scanner

Din partiţia obţinută cu relaţiile (7.47) şi (7.49)

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} = \begin{bmatrix} 1 & -y(N-1) & \varphi_n^{\mathrm{T}}(N-1) \\ -y(N-1) & \rho_0(N-1) & -r_n^{\mathrm{T}}(N-1) \\ \varphi_n(N-1) & -r_n(N-1) & R_n(N-1) \end{bmatrix}, \tag{7.157}$$

rezultă imediat identităţile:

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N-1) \end{bmatrix} = \begin{bmatrix} 1-\gamma_n^*(N-1) \\ -\varepsilon_n^f(N-1) \\ 0_n \end{bmatrix}, \tag{7.158}$$

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \theta_n(N-1) \end{bmatrix} = \begin{bmatrix} -\varepsilon_n^f(N-1) \\ \sigma_n(N-1) \\ 0_n \end{bmatrix}, \tag{7.159}$$

din combinarea cărora rezultă

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N) \end{bmatrix} \left\{ \begin{bmatrix} 1 \\ 0 \\ -K_n^*(N-1) \end{bmatrix} + \frac{\varepsilon_n^f(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 0 \\ 1 \\ \theta_n(N-1) \end{bmatrix} \right\} =$$

$$= \begin{bmatrix} 1-\gamma_n^*(N-1) - \frac{\varepsilon_n^{f2}(N-1)}{\sigma_n(N-1)} \\ 0_{n+1} \end{bmatrix} \tag{7.160}$$

şi comparând cu (7.147) rezultă (7.141) şi (7.145).

Din partiţia similară cu (7.157)

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} = \begin{bmatrix} 1 & -y(N-1) & \varphi_n^{\mathrm{T}}(N-1) \\ -y(N-1) & \rho_0(N-2) & -r_n^{\mathrm{T}}(N-2) \\ \varphi_n(N-1) & -r_n(N-2) & R_n(N-2) \end{bmatrix}, \tag{7.161}$$

rezultă după calcule imediate:

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -K_n(N-1) \end{bmatrix} = \begin{bmatrix} 1-\gamma_n(N-1) \\ -e_n^f(N-1) \\ 0_n \end{bmatrix}, \tag{7.162}$$

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \theta_n(N-2) \end{bmatrix} = \begin{bmatrix} -e_n^f(N-1) \\ \sigma_n(N-2) \\ 0_n \end{bmatrix}, \tag{7.163}$$

care pot fi combinate, rezultând

$$\begin{bmatrix} 1 & \varphi_{n+1}^{\mathrm{T}}(N) \\ \varphi_{n+1}(N) & R_{n+1}(N-1) \end{bmatrix} \left\{ \begin{bmatrix} 1 \\ 0 \\ -K_n(N-1) \end{bmatrix} + \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 0 \\ 1 \\ \theta_n(N-2) \end{bmatrix} \right\} =$$

$$= \begin{bmatrix} 1-\gamma_n(N-1) - \frac{e_n^{f2}(N-1)}{\sigma_n(N-2)} \\ 0_{n+1} \end{bmatrix} \tag{7.164}$$

şi comparând cu (7.148) rezultă (7.142) şi (7.146).

Scanned with OKEN Scanner

7.3.3. Variante ale algorimilor latice a erorilor de predicţie şi predictorilor

Algoritmul 7.2.1. propagă în timp, cu un cost de $\mathcal{O}(n_{max})$ operaţii/ pas de eşantionare, următoarele mărimi: parametrii latice (coeficienţii de reflexie) şi erorile de predicţie apriori şi aposteriori.

Utilizând numai aceste mărimi, pot fi deduşi predictori la un moment oarecare de timp N prin ecuaţiile (7.123) şi (7.131) din Tabelul 7.6. (în care au fost grupate toate ecuaţiile recursive în ordin pentru mai uşoara lor referire în capitolele următoare) şi prin ecuaţia (7.124) transformată astfel încât să devină o recursie "pură" în ordin. Din relaţiile (7.94),(7.95), (7.99)–(7.101) rezultă

$$\theta_n(N-1) = \theta_n(N) - K_n^*(N)e_n^f(N), \tag{7.165}$$

$$\theta_n^b(N-1) = \theta_n^b(N) - K_n^*(N)e_n^b(N), \tag{7.166}$$

care înlocuite în (7.124) conduc la

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma(N-1)} \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} - \\ - (e_n^b(N) - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} e_n^f(N)) \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix}. \tag{7.167}$$

Relaţiile (7.123), (7.167) şi (7.131) realizează calculul predictorilor $\theta_n(N), \theta_n^b(N)$ şi $K_n^*(N)$ recursiv în ordin. Iniţializarea acestor recursii la ordinul $n = 1$ rezultă direct din (7.83),(7.84) şi (7.90). În Tabelul 7.7. este prezentat pasul 2.3 pentru completarea Algoritmului 7.2.1. pentru ca acesta să devină şi algoritm al predictorilor.

O altă variantă a algoritmului 7.2.1. se poate obţine prin utilizarea vectorului amplificărilor Kalman apriori din (7.132), înlocuind $\theta_n^b(N-1)$ cu

$$\theta_n^b(N-1) = \theta_n^b(N) - K_n(N)\varepsilon_n^b(N),$$

iar $\theta_n(N-1)$ prin (7.94).

Varianta cea mai frecvent utilizată a algoritmului latice a erorilor de predicţie şi predictorilor este prezentată în Tabelul 7.8. În secţiunea latice există un singur filtru latice, cel al erorilor de predicţie aposteriori, în plus faţă de Algoritmul 7.2.1. fiind iterat şi raportul aposteriori/apriori $\gamma_n^0(N)$ (utilizând (7.135) şi (7.102) rezultă ecuaţia recursivă în ordin pentru $\gamma_n^0(N)$).

Pasul 2.2, care se execută obligatoriu la fiecare moment de eşantionare, necesită în Algoritmul 7.3.1. $10n_{max}$ înmulţiri sau împărţiri şi $5n_{max}$ adunări, faţă de $10n_{max}$ înmulţiri sau împărţiri şi $6n_{max}$ adunări, câte erau necesare în Algoritmul 7.2.1.

7.3.4. Iniţializarea exactă a algoritmilor latice

Algoritmii latice posedă o proprietate remarcabilă, care în general nu este prezentă la alţi algoritmi recursivi în timp: nu este necesară o altă procedură pentru a realiza o iniţializare exactă. Prin iniţializare exactă vom înţelege iniţializarea la momentul $N = 0$ a mărimilor care intervin în algoritm, astfel încât după un număr de paşi, atunci când soluţia problemei CMMP (dată, de exemplu, de predictorul $\theta_n(N)$ sau predicţia $\hat{y}_n(N)$) devine unică, algoritmul să furnizeze exact acea soluţie.

În partea a treia a acestei lucrări vom constata că algoritmii rapizi recursivi în timp nu pot fi iniţializaţi exact decât utilizând alte proceduri, care au o complexitate de calcul mult mai mare.

Scanned with OKEN Scanner

Algoritmul 7.3.1.

ALGORITM LATICE A ERORILOR DE PREDICTIE SI A PREDICTORILOR		*Num. op.* ⋆	/	+
1. *Iniţializare la momentul* $(N = 0)$				
Pentru $n = 0, \ldots, n_{max} - 1 \quad \alpha_n(0) = 0$				
$\sigma_0(0) = \sigma_0^b(0) = 0$				
2. *Recursii în timp*				
Pentru $N = 1, 2, \ldots, N_{max}$				
2.1 *Iniţializare recursii în ordin*				
$\varepsilon_0^f(N) = y(N); \quad \varepsilon_0^b(N) = y(N-1)$				
$\sigma_0(N) = \sigma_0(N-1) + \varepsilon_0^{f2}(N); \quad \sigma_0^b(N) = \sigma_0(N-1)$		1	0	1
$\gamma_0^0(N) = 1$				
2.2 *Recursii în ordin:*				
Pentru $n = 0, \ldots, n_{max} - 1$				
$\alpha_n(N) = \alpha_n(N-1) + \varepsilon_n^f(N)\varepsilon_n^b(N)/\gamma_n^0(N)$	(1)	1	1	1
$K_{n+1}^f(N) = -\alpha_n(N)/\sigma_n^b(N) \quad (\star)$	(2)	0	1	0
$K_{n+1}^b(N) = -\alpha_n(N)/\sigma_n(N) \quad (\star)$	(3)	0	1	0
$\sigma_{n+1}(N) = \sigma_n(N)(1 - K_{n+1}^f(N)K_{n+1}^b(N))$	(4)	2	0	1
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1)(1 - K_{n+1}^f(N-1)K_{n+1}^b(N-1))$	(5)	1	0	0
$\varepsilon_{n+1}^f(N) = \varepsilon_n^f(N) + K_{n+1}^f(N)\varepsilon_n^b(N)$	(6)	1	0	1
$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) + K_{n+1}^b(N-1)\varepsilon_n^f(N-1)$	(7)	1	0	1
$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \varepsilon_n^{b2}(N)/\sigma_n^b(N) \quad \star$	(8)	1	0	1
2.3 *Dacă se doreşte evaluarea predictorilor la momentul N*				
Atunci:				
2.3.1 Iniţializare recursii în ordin:				
$\theta_1(N) = K_1^f(N); \quad \theta_1^b(N) = K_1^b(N-1)$				
$K_1^*(N) = -\varepsilon_0^b(N)/\sigma_0^b(N)$	(9)		1	
2.3.2 Recursii în ordin:				
Pentru $n = 1, \ldots, n_{max} - 1$				
$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + K_{n+1}^f(N) \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$	(10)	n	0	n
$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} + K_{n+1}^b(N-1) \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} - \frac{(\varepsilon_n^b(N) + K_{n+1}^b(N-1)\varepsilon_n^f(N))}{\gamma_n^0(N)} \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix}$	(11)	$2n+1$	1	$2n+1$
$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$	(12)	n	1	n
$(\star)$: *Dacă oricare dintre împărţitorii* $\sigma_n(N), \sigma_n^b(N)$ *sau* $\gamma_n^0(N)$ *este nul, rezultatul împărţirii respective se setează la 0.*				

Tabelul 7.8. Algoritm latice a erorilor de predicţie şi predictorilor optimali.

Scanned with OKEN Scanner

Problema iniţializării exacte este strâns legată de tratarea cazului singularităţii matricei $R_n(N)$. Conform ipotezei *fereastră anterioară*, toate valorile $y(t)$ sunt nule pentru $t \leq 0$, deci $R_n(N)$ este evident singulară pentru valori mici ale lui N. În lema următoare se analizează relaţia dintre singularitatea matricei $R_n(N)$ şi numărul de paşi trecuţi de la apariţia primei valori nenule $y(s)$.

Lema 7.3.2. *Fie s momentul primei apariţii a unei valori nenule $y(s) \neq 0$, $y(t) = 0$ pentru toţi $t < s$.*

Determinantul matricei $R_n(t)$ satisface

$$|R_n(t)| \begin{cases} = 0 & \text{pentru } t < s+n; \\ > 0 & \text{pentru } t \geq s+n. \end{cases} \tag{7.168}$$

Demonstraţie. i) pentru ordinul $n = 1$ proprietatea este imediată:

$$R_1(t) = \sum_{i=0}^{t} y^2(i-1) = \begin{cases} 0 & \text{pentru } t \leq s; \\ y^2(s) & \text{pentru } t = s+1; \\ y^2(s) + \sum_{i=s+1}^{t-1} y^2(i) & \text{pentru } t > s+1. \end{cases} \tag{7.169}$$

ii) Pentru ordinul n demonstrăm mai întâi nesingularitatea pentru $t \geq s+n$, procedând prin reducere la absurd. Presupunem $|R_n(t)| = 0$ pentru $t \geq s+n$. Rezultă că există $z = \begin{bmatrix} z_1 & z_2 & \dots & z_n \end{bmatrix}^{\mathrm{T}}$, $z \neq 0$ cu $R_n(t)z = 0$. Rezultă

$$z^{\mathrm{T}} R_n(t) z = z^{\mathrm{T}} \sum_{i=0}^{t} \varphi_n(i)\varphi_n^{\mathrm{T}}(i) z = \sum_{i=0}^{t} (\varphi_n^{\mathrm{T}}(i)z)^2 = 0,$$

deci $\varphi_n(i)^{\mathrm{T}} z = 0$ pentru $i = 0, \dots, t$. Scriind aceste egalităţi pentru $i = s+1, \dots, s+n$ se obţine

$$\begin{aligned} z_1 y(s) &= 0 \Rightarrow z_1 = 0, \\ z_1 y(s+1) + z_2 y(s) &= 0 \Rightarrow z_2 = 0, \\ z_1 y(s+n-1) + z_2 y(s+n-2) + \dots + z_n y(s) &= 0 \Rightarrow z_n = 0, \end{aligned}$$

adică $z \equiv 0$, ceea ce contrazice ipoteza $|R_n(t)| = 0$ pentru $t \geq s+n$.

iii) Demonstrăm că $|R_n(N)| = 0$ implică $|R_{n+1}(N)| = 0$. Fie $z = \begin{bmatrix} z_1 & z_2 & \dots & z_n \end{bmatrix}^{\mathrm{T}}$, $z \neq 0$ astfel încât $R_n(N)z = 0$, sau echivalent

$$\sum_{i=0}^{N} (\varphi_n^{\mathrm{T}}(i)z)^2 = 0. \tag{7.170}$$

Definim $w = \begin{bmatrix} z \\ 0 \end{bmatrix}$. Avem $w \neq 0$ şi

$$\sum_{i=0}^{N} (\varphi_{n+1}^{\mathrm{T}}(i)w)^2 = \sum_{i=0}^{N} (\varphi_n^{\mathrm{T}}(i)z)^2 = 0, \tag{7.171}$$

de unde rezultă $w^{\mathrm{T}} R_{n+1}(N) w = 0$ şi datorită pozitiv-semidefinirii matricei $R_{n+1}(N)$ rezultă $|R_{n+1}(N)| = 0$.

Scanned with OKEN Scanner

iv) Demonstrăm că $|R_n(N)| = 0$ implică $|R_{n+1}(N+1)| = 0$.

Fie $z \neq 0$ a.î. $R_n(N)z = 0$ ca mai sus. Definim $w = \begin{bmatrix} 0 \\ z \end{bmatrix}$. Avem $w \neq 0$ şi

$$\sum_{i=0}^{N}(\varphi_n^{\mathrm{T}}(i)z)^2 = \sum_{i=0}^{N}(\varphi_{n+1}(i+1)^{\mathrm{T}}w)^2 = \sum_{i=0}^{N+1}(\varphi_{n+1}^{\mathrm{T}}(i)w)^2 = \\ = w^{\mathrm{T}}R_{n+1}(N+1)w = 0 \tag{7.172}$$

şi rezultă datorită pozitiv semidefinirii matricei $R_{n+1}(N+1)$ că determinantul ei este nul.

v) Ţinând cont de iii) şi iv), prin iteraţii succesive plecând de la ordinul $n = 1$, pentru care avem relaţia (7.169), rezultă $|R_n(t)| = 0$ pentru $t < s + n$, deci (7.168) e complet demonstrată. □

Soluţiile problemei de modelare CMMP în cazul singularităţii matricei $R_n(N)$ sunt considerate în lema următoare.

Lema 7.3.3. *Fie N astfel încât*

$$s + 1 \leq N < s + n, \tag{7.173}$$

unde s este primul moment pentru care $y(s) \neq 0$. Conform Lemei 7.3.2., $|R_n(N)| = 0$ şi problema CMMP nu are soluţie unică.

i) Predictorul de normă minimă care minimizează $V_n(\theta_n)$ (relaţia (7.32)) este

$$\theta_n(N) = \begin{bmatrix} \theta_{N-s}(N) \\ 0_{n-N-s} \end{bmatrix}, \tag{7.174}$$

iar valoarea minimă a criteriului $V_n(\theta_n)$ este

$$\sigma_n(N) = y^2(s). \tag{7.175}$$

ii) Predictorul de normă minimă care minimizează $V_n^b(\theta^b)$ (relaţia (7.33)) este

$$\theta_n^b(N) = 0_n \tag{7.176}$$

pentru care valoarea criteriului este

$$\sigma_n^b(N) = 0. \tag{7.177}$$

iii) Valoarea intercorelaţiilor de eşantion $\alpha_n(N)$ pentru n şi N ca în (7.173) este

$$\alpha_n(N) = 0. \tag{7.178}$$

Demonstraţie. i) Notând $\theta_n = \begin{bmatrix} \theta^1 & \theta^2 & \ldots \theta^n \end{bmatrix}^{\mathrm{T}}$, criteriul (7.32) se scrie

$$V_n(\theta_n) = y^2(s) + (y(s+1) + \theta^1 y(s))^2 + \ldots + (y(N) + \theta^1 y(N-1) + \ldots + \theta^{N-s}y(s))^2 = \\ = V_{N-s}(\theta_{N-s}). \tag{7.179}$$

Rezultă imediat că valoarea minimă a criteriului este $y^2(s)$ şi se obţine, datorită nesingularităţii matricei $R_{N-s}(N)$, pentru

$$\theta_n = \begin{bmatrix} \theta_{N-s} \\ X \end{bmatrix}, \tag{7.180}$$

cu $X \in \mathcal{R}^{n-N+s}$ vector arbitrar.

Predictorul de normă minimă de tipul (7.180) este evident dat de (7.174).

ii) Notând $\theta_n^b = \left[\theta^1 \ \ldots \ \theta^n \right]^{\mathrm{T}}$, criteriul (7.33) se scrie

$$V_n^b(\theta_n^b) = \sum_{t=0}^{N}(y(t-n-1)+\theta^1 y(t-1)+\ldots+\theta^n y(t-n))^2 = \tag{7.181}$$
$$= (\theta^1 y(s))^2 + (\theta^1 y(s+1)+\theta^2 y(s))^2 + (\theta^1 y(N-1)+\ldots+\theta^{N-s-1} y(s))^2.$$

Rezultă imediat că (7.176) minimizează (7.181) care are ca valoare minimă (7.177).

iii) Conform definiţiei (7.59), intercorelaţia erorilor de predicţie înainte şi înapoi corespunzătoare predictorilor de normă minimă (7.174) şi (7.176) va fi

$$\alpha_n(N) = \rho_{n+1}(N) - r_n^{\mathrm{T}}(N)\theta_n^b(N) = \rho_{n+1}(N) = \sum_{t=0}^{N} y(t)y(t-n-1) = 0$$

pentru valorile n şi N care satisfac (7.173).

□

Din lema precedentă pot fi deduse măsurile care trebuie luate pentru a putea itera algoritmul latice în cazul în care matricea $R_n(N)$ este singulară (singularitate reflectată în algoritm prin $\sigma_n(N) = 0$ şi $\sigma_n^b(N) = 0$). Analizând ecuaţiile algoritmilor latice pentru calculul erorilor de predicţie şi pentru predictori se observă că se pot urma două variante:

a) înlocuirea inverselor $(\sigma_n(\cdot))^{-1}$ şi $(\sigma_n^b(\cdot))^{-1}$ cu pseudoinverse (deci în cazul scalar, cu orice număr real), variantă introdusă pentru cazul multivariabil în [64], care conduce la propagarea prin algoritm, cât timp matricea $R_n(N)$ este singulară, a unor soluţii ale problemei CMMP, iar în cazul în care $R_n(N)$ devine nesingulară, se propagă în continuare soluţiile unice CMMP, deci iniţializarea algoritmului este exactă;

b) înlocuirea inverselor $(\sigma_n(\cdot))^{-1}$ şi $(\sigma_n^b(\cdot))^{-1}$ cu valoarea zero, pentru care se obţine, în plus faţă de cazul a), ca soluţiile propagate să fie chiar cele de normă minimă, ceea ce prezintă un evident avantaj numeric.

Scanned with OKEN Scanner

Capitolul 8

Algoritmi Latice Normalizată pentru Modele AR Monovariabile

8.1. Algoritmi latice pentru erorile de predicţie normalizate

Performanţele numerice ale algoritmilor latice prezentaţi în capitolul anterior sunt în mod decisiv determinate de proprietăţile secţiunii latice (care propagă în timp erorile de predicţie şi coeficienţii de reflexie) pentru că la nivelul acestei secţiuni se poate produce acumularea în timp a erorilor datorite preciziei finite a operaţiilor aritmetice.

Normalizarea secţiunii latice care va fi prezentată în acest capitol va conduce la obţinerea unor performanţe numerice superioare faţă de laticea nenormalizată atât în ceea ce priveşte numărul de operaţii necesar, cât şi în privinţa sensibilităţii la erorile produse de operaţii aritmetice pe un număr finit de biţi. Preţul plătit pentru aceste performanţe este însă pierderea semnificaţiei (foarte utile în aplicaţiile de predicţie) mărimilor propagate în timp în secţiunea latice (erori de predicţie, dispersii ale erorilor, direct utilizabile în procedurile de estimare a ordinului). În aceste condiţii, rezolvarea Problemei 7.2.1. prin algoritmi normalizaţi implică un număr suplimentar de calcule pentru trecerea de la mărimile normalizate la cele nenormalizate.

Definim în continuare următoarele mărimi:

$$\tilde{\varepsilon}_n^f(N) = \frac{\varepsilon_n^f(N)}{\sqrt{\sigma_n(N)}}, \tag{8.1}$$

$$\tilde{\varepsilon}_n^b(N) = \frac{\varepsilon_n^b(N)}{\sqrt{\sigma_n^b(N)}}, \tag{8.2}$$

$$\overline{\alpha}_n(N) = \frac{\alpha_n(N)}{\sqrt{\sigma_n(N)\sigma_n^b(N)}}, \tag{8.3}$$

care rezultă prin normarea erorilor de predicţie în raport cu "dispersiile de eşantion" corespunzătoare (defiiniţiile sunt valide pentru $\sigma_n(N)$ şi $\sigma_n^b(N)$ nenule, ceea ce se întâmplă pentru $N > s + n$, conform Lemelor 7.3.2. şi 7.3.3.).

Lema 8.1.1. *Mărimile normate definite în (8.1)-(8.3) sunt subunitare în modul:*

$$|\tilde{\varepsilon}_n^f(N)| \leq 1, \tag{8.4}$$

$$|\tilde{\varepsilon}_n^b(N)| \leq 1, \tag{8.5}$$

$$|\overline{\alpha}_n(N)| \leq 1. \tag{8.6}$$

Demonstraţie. Din definiţiile (7.39),(7.41) rezultă:

$$\sigma_n(N) = \sum_{t=0}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))^2 = \varepsilon_n^{f2}(N) + \sum_{t=0}^{N-1}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))^2, \tag{8.7}$$

$$\sigma_n^b(N) = \sum_{t=0}^{N}(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N))^2 =$$

$$= \varepsilon_n^{b2}(N) + \sum_{t=0}^{N-1}(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N))^2, \tag{8.8}$$

iar din aceste două relaţii se deduc imediat (8.4) şi (8.5).

Mărimea $\alpha_n(N)$ are semnificaţia "intercorelaţie de eşantion" a erorilor de predicţie corespunzătoare modelelor optimale $\theta_n(N)$ şi $\theta_n^b(N)$:

$$\sum_{t=0}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N)) = \rho_{n+1}(N) - r_n^{\mathrm{T}}(N)\theta_n^b(N) -$$

$$-r_n^{b\mathrm{T}}(N)\theta_n(N) + \theta_n^{\mathrm{T}}(N)R_n(N)\theta_n^b(N) = \rho_{n+1}(N) - r_n^{\mathrm{T}}(N)\theta_n^b(N) = \alpha_n(N). \tag{8.9}$$

Scriind inegalitatea lui Schwartz

$$\sum_{t=0}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))^2 \sum_{t=0}^{N}(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N))^2 \geq$$

$$\geq \left\{\sum_{t=0}^{N}(y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))(y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N))\right\}^2, \tag{8.10}$$

rezultă

$$\sigma_n(N)\sigma_n^b(N) \geq \alpha_n^2(N).$$

Egalitatea în (8.10) se obţine în cazul în care există $\lambda \neq 0$ astfel încât

$$y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N) = [y(t-n-1) - \varphi_n^{\mathrm{T}}(t)\theta_n^b(N)]\lambda, \quad \forall t = 0, \ldots, N.$$

În condiţiile ipotezei "fereastră anterioară" această relaţie nu poate avea loc pentru $N \geq s$ unde s este primul moment pentru care $y(s) \neq 0$ (pentru că la $t = s$, $y(t) \neq 0$, $\varphi_n(t) = 0$, $y(t-n-1) = 0$ şi egalitatea de mai sus nu poate avea loc pentru nici o valoare $\lambda \neq 0$).În acest mod relaţia (8.6) este complet demonstrată.

□

Relaţiile (8.4)-(8.6) asigură că mărimile normate pot fi reprezentate în virgulă fixă, ca numere subunitare, aceasta conducând la posibilitatea scăderii sensibile a timpului necesar pentru efectuarea operaţiilor aritmetice (prin utilizarea unor biblioteci aritmetice în virgulă fixă, mult mai rapide decât cele în virgulă mobilă).

Lema 8.1.2. *Prin înlocuirea în secţiunea latice (pasul 2.2) din Algoritmul 7.3.1. a mărimilor nenormalizate cu mărimile normalizate (8.1)-(8.3), se obţine următoarea secţiune latice:*

$$\bar{\varepsilon}_{n+1}^f(N) = (1 - \overline{\alpha}_n^2(N))^{-1/2}(\bar{\varepsilon}_n^f(N) - \overline{\alpha}_n(N)\bar{\varepsilon}_n^b(N)), \tag{8.11}$$

Scanned with OKEN Scanner

$$\tilde{\varepsilon}^b_{n+1}(N) = (1-\overline{\alpha}^2_n(N-1))^{-1/2}(\tilde{\varepsilon}^b_n(N-1)-\overline{\alpha}_n(N-1)\tilde{\varepsilon}^f_n(N-1)), \tag{8.12}$$

$$\overline{\alpha}_n(N) = \overline{\alpha}_n(N-1)\left(1-\frac{\tilde{\varepsilon}^{f2}_n(N)}{\gamma^0_n(N)}\right)^{1/2}\left(1-\frac{\tilde{\varepsilon}^{b2}_n(N)}{\gamma^0_n(N)}\right)^{1/2}+\frac{\tilde{\varepsilon}^f_n(N)\tilde{\varepsilon}^b_n(N)}{\gamma^0_n(N)}, \tag{8.13}$$

$$\gamma^0_{n+1}(N) = \gamma^0_n(N)-\tilde{\varepsilon}^{b2}_n(N). \tag{8.14}$$

Demonstraţie. Înlocuind relaţiile (8.1)–(8.3) în ecuaţiile (6) şi (7) ale Algoritmului 7.3.1. rezultă

$$\tilde{\varepsilon}^f_{n+1}(N)\sqrt{\sigma_{n+1}(N)} = \tilde{\varepsilon}^f_n(N)\sqrt{\sigma_n(N)}-\frac{\alpha_n(N)\sqrt{\sigma_n(N)\sigma^b_n(N)}}{\sigma^b_n(N)}\tilde{\varepsilon}^b_n(N)\sqrt{\sigma^b_n(N)}, \tag{8.15}$$

sau

$$\tilde{\varepsilon}^f_{n+1}(N) = \sqrt{\frac{\sigma_n(N)}{\sigma_{n+1}(N)}}(\tilde{\varepsilon}^f_n(N)-\overline{\alpha}_n(N)\tilde{\varepsilon}^b_n(N)) \tag{8.16}$$

şi

$$\tilde{\varepsilon}^b_{n+1}(N+1)\sqrt{\sigma^b_{n+1}(N+1)} = \tilde{\varepsilon}^b_n(N)\sqrt{\sigma^b_n(N)}- -\frac{\overline{\alpha}_n(N)\sqrt{\sigma_n(N)\sigma^b_n(N)}}{\sigma_n(N)}\varepsilon^f_n(N)\sqrt{\sigma_n(N)}, \tag{8.17}$$

$$\tilde{\varepsilon}^b_{n+1}(N+1) = \sqrt{\frac{\sigma^b_n(N)}{\sigma^b_{n+1}(N)}}(\tilde{\varepsilon}^b_n(N)-\overline{\alpha}_n(N)\tilde{\varepsilon}^f_n(N)). \tag{8.18}$$

Din ecuaţia (4) din Tabelul 7.7 rezultă

$$\frac{\sigma_{n+1}(N)}{\sigma_n(N)} = 1-\frac{\alpha^2_n(N)}{\sigma_n(N)\sigma^b_n(N)} = 1-\overline{\alpha}^2_n(N), \tag{8.19}$$

iar din ecuaţia (5) rezultă

$$\frac{\sigma^b_{n+1}(N+1)}{\sigma^b_n(N)} = 1-\frac{\alpha^2_n(N)}{\sigma_n(N)\sigma^b_n(N)} = 1-\overline{\alpha}^2_n(N). \tag{8.20}$$

Înlocuind (8.19) şi (8.20) în (8.16) şi (8.18) rezultă imediat primele două ecuaţii ale secţiunii latice transformate, (8.11) şi (8.12).

Introducând mărimile normate, ecuaţia (1) din Algoritmul 7.3.1. se poate scrie

$$\overline{\alpha}_n(N)\sqrt{\sigma_n(N)\sigma^b_n(N)} = \overline{\alpha}_n(N-1)\sqrt{\sigma_n(N-1)\sigma^b_n(N-1)}+ +\frac{\tilde{\varepsilon}^f_n(N)\tilde{\varepsilon}^b_n(N)\sqrt{\sigma_n(N)\sigma^b_n(N)}}{\gamma^0_n(N)}, \tag{8.21}$$

sau

$$\overline{\alpha}_n(N) = \overline{\alpha}_n(N-1)\sqrt{\frac{\sigma_n(N-1)}{\sigma_n(N)}\frac{\sigma^b_n(N-1)}{\sigma^b_n(N)}}+\frac{\tilde{\varepsilon}^f_n(N)\tilde{\varepsilon}^b_n(N)}{\gamma^0_n(N)}. \tag{8.22}$$

Din ecuaţiile (7.96) şi (7.100) rezultă

$$\frac{\sigma_n(N-1)}{\sigma_n(N)} = 1-\frac{\varepsilon^{f2}_n}{\gamma^0_n(N)\sigma_n(N)} = 1-\frac{\tilde{\varepsilon}^{f2}_n(N)}{\gamma^0_n(N)}, \tag{8.23}$$

iar din ecuaţiile (7.97) şi (7.101)

$$\frac{\sigma_n^b(N-1)}{\sigma_n^b(N)} = 1 - \frac{\varepsilon_n^{b2}(N)}{\gamma_n^0(N)\sigma_n^b(N)} = 1 - \frac{\bar{\varepsilon}_n^{b2}(N)}{\gamma_n^0(N)}. \tag{8.24}$$

Utilizând (8.23) şi (8.24) în relaţia (8.21) rezultă ecuaţia (8.13) din secţiunea latice transformată.

Din ecuaţia (7.135) şi ecuaţia (7.102) rezultă

$$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N)} = \gamma_n^0(N) - \bar{\varepsilon}_n^{b2}(N). \tag{8.25}$$

□

Se observă că în secţiunea latice transformată din Lema 8.1.2. nu mai apar ecuaţii recursive pentru calculul dispersiilor, acestea nemaifiind necesare datorită efectuării operaţiei de normare.

O formă şi mai compactă a secţiunii latice se poate obţine eliminând şi ecuaţia recursivă de calcul pentru $\gamma_n^0(N)$, printr-o normalizare suplimentară a erorilor de predicţie

$$\bar{\varepsilon}_n^f(N) = \frac{\tilde{\varepsilon}_n^f(N)}{\gamma_n^0(N)}, \tag{8.26}$$

$$\bar{\varepsilon}_n^b(N) = \frac{\tilde{\varepsilon}_n^b(N)}{\gamma_n^0(N)}. \tag{8.27}$$

Cu aceste definiţii, relaţia (8.13) devine

$$\bar{\alpha}_n(N) = \bar{\alpha}_n(N-1)(1-\bar{\varepsilon}_n^{f2}(N))^{1/2}(1-\bar{\varepsilon}_n^{b2}(N))^{1/2} + \bar{\varepsilon}_n^f(N)\bar{\varepsilon}_n^b(N), \tag{8.28}$$

iar relaţia (8.11) se scrie

$$\bar{\varepsilon}_{n+1}^f(N)\sqrt{\gamma_{n+1}^0(N)} = (1-\bar{\alpha}_n^2(N))^{-1/2}(\bar{\varepsilon}_n^f(N) - \bar{\alpha}_n(N)\bar{\varepsilon}_n^b(N))\sqrt{\gamma_n^0(N)}. \tag{8.29}$$

Din relaţia (8.25) rezultă

$$\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N)} = 1 - \frac{\bar{\varepsilon}_n^{b2}(N)}{\gamma_n^0(N)} = 1 - \bar{\varepsilon}_n^{b2}(N), \tag{8.30}$$

deci (8.29) se scrie

$$\bar{\varepsilon}_{n+1}^f(N) = (1-\bar{\varepsilon}_n^{b2}(N))^{-1/2}(1-\bar{\alpha}_n^2(N))^{-1/2}(\bar{\varepsilon}_n^f(N) - \bar{\alpha}_n(N)\bar{\varepsilon}_n^b(N)). \tag{8.31}$$

Înlocuim erorile renormalizate în (8.12)

$$\bar{\varepsilon}_{n+1}^b(N)\sqrt{\gamma_{n+1}^0(N)} = (1-\bar{\alpha}_n^2(N-1))^{-1/2}(\bar{\varepsilon}_n^b(N-1) - \bar{\alpha}_n(N-1)\bar{\varepsilon}_n^f(N-1))\sqrt{\gamma_n^0(N-1)}, \tag{8.32}$$

iar din (7.137) şi (7.102) rezultă

$$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \frac{\varepsilon_n^{f2}(N-1)}{\sigma_n(N-1)} = \gamma_n^0(N-1)(1-\bar{\varepsilon}_n^{f2}(N-1)), \tag{8.33}$$

deci

$$\bar{\varepsilon}_{n+1}^b(N) = (1-\bar{\varepsilon}_n^{f2}(N-1))^{-1/2}(1-\bar{\alpha}_n^2(N-1))^{-1/2}(\bar{\varepsilon}_n^b(N-1) - \bar{\alpha}_n(N-1)\bar{\varepsilon}_n^f(N-1)). \tag{8.34}$$

Scanned with OKEN Scanner

Lema 8.1.3. *Au loc inegalităţile:*

$$\gamma_n^0(N) > 0, \tag{8.35}$$
$$|\overline{\varepsilon}_n^f(N)| \leq 1, \tag{8.36}$$
$$|\overline{\varepsilon}_n^b(N)| \leq 1, \tag{8.37}$$

asigurând buna definire a mărimilor normalizate şi a calculelor din ecuaţiile recursive în ordin ale erorilor normalizate.

Demonstraţie. Relaţiile (8.23) şi (8.24) se pot scrie:

$$1 - \overline{\varepsilon}_n^{f2}(N) = \frac{\sigma_n(N-1)}{\sigma_n(N)}, \tag{8.38}$$
$$1 - \overline{\varepsilon}_n^{b2}(N) = \frac{\sigma_n^b(N-1)}{\sigma_n^b(N)}. \tag{8.39}$$

Din relaţiile (8.7) şi (8.8) rezultă că dispersiile de eşantion $\sigma_n(\cdot)$ şi $\sigma_n^b(\cdot)$ sunt pozitive iar pentru $N \geq s+n+1$, matricea $R_{n+1}(N)$ este nesingulară (conform Lemei 7.3.2.), aceasta implicând din (7.53), (7.54) că $\sigma_n(N) \neq 0$ şi $\sigma_n^b(N) \neq 0$. În consecinţă, din (8.38) şi (8.39) se obţin inegalităţile (8.36) şi (8.37).

Din relaţiile (8.30) şi (8.39) rezultă

$$\begin{aligned}\gamma_{n+1}^0(N) &= \gamma_n^0(N)(1-\overline{\varepsilon}_n^{b2}(N)) = \gamma_{n-1}^0(N)(1-\overline{\varepsilon}_{n-1}^{b2}(N))(1-\overline{\varepsilon}_n^{b2}(N)) = \\ &= \ldots = (1-\overline{\varepsilon}_0^{b2}(N))\ldots(1-\overline{\varepsilon}_n^{b2}(N)) \geq 0.\end{aligned}$$

□

Introducând notaţia

$$F(a,b,c) = (1-b^2)^{-1/2}(a-bc)(1-c^2)^{-1/2}, \tag{8.40}$$

observăm că relaţiile (8.31) şi (8.34) se pot scrie:

$$\overline{\varepsilon}_{n+1}^f(N) = F(\overline{\varepsilon}_n^f(N), \overline{\alpha}_n(N), \overline{\varepsilon}_n^b(N)), \tag{8.41}$$
$$\overline{\varepsilon}_{n+1}^b(N) = F(\overline{\varepsilon}_n^b(N-1), \overline{\alpha}_n(N-1), \overline{\varepsilon}_n^f(N-1)). \tag{8.42}$$

Relaţia (8.28) poate fi rescrisă

$$\overline{\alpha}_n(N-1) = (1-\overline{\varepsilon}_n^{f2}(N))^{-1/2}(\overline{\alpha}_n(N) - \overline{\varepsilon}_n^f(N)\overline{\varepsilon}_n^b(N))(1-\overline{\varepsilon}_n^{b2}(N))^{-1/2}, \tag{8.43}$$

sau

$$\overline{\alpha}_n(N-1) = F(\overline{\alpha}_n(N), \overline{\varepsilon}_n^f(N), \overline{\varepsilon}_n^b(N)). \tag{8.44}$$

Definim "inversa în raport cu variabila a" a relaţiei (8.40)

$$\overline{F}(d,b,c) = (1-b^2)^{1/2}(1-c^2)^{1/2}d + bc \tag{8.45}$$

şi va rezulta

$$\overline{\alpha}_n(N) = \overline{F}(\overline{\alpha}_n(N-1), \overline{\varepsilon}_n^f(N), \overline{\varepsilon}_n^b(N)). \tag{8.46}$$

Scanned with OKEN Scanner

Algoritmul 8.1.1.

ALGORITM LATICE PENTRU ERORILE DE PREDICTIE NORMALIZATE		*Num. op.* ⋆ / + √
1. *Iniţializare la momentul* $(N = 0)$		
Pentru $n = 0, \ldots, n_{max} - 1 \quad \overline{\alpha}_n(0) = 0$		
$\sigma_0(0) = 0$		
2. *Recursii în timp*		
Pentru $N = 1, 2, \ldots, N_{max}$		
2.1 *Iniţializare recursii în ordin*		
$\sigma_0(N) = \sigma_0(N-1) + y^2(N)$	(1)	1 0 1 0
$\overline{\varepsilon}_0^f(N) = (\sigma_0(N))^{-1/2} y(N)$	(2)	0 1 0 1
$\overline{\varepsilon}_0^b(N) = \overline{\varepsilon}_0^f(N-1)$		
2.2 *Recursii în ordin:*		
Pentru $n = 0, \ldots, n_{max} - 1$		
$\overline{\alpha}_n(N) = \overline{F}(\overline{\alpha}_n(N-1), \overline{\varepsilon}_n^f(N), \overline{\varepsilon}_n^b(N))$	(3)	5 0 3 2
$\overline{\varepsilon}_{n+1}^f(N) = F(\overline{\varepsilon}_n^f(N), \overline{\alpha}_n(N), \overline{\varepsilon}_n^b(N))$	(4)	2 0 1 0
$\overline{\varepsilon}_{n+1}^b(N) = F(\overline{\varepsilon}_n^b(N-1), \overline{\alpha}_n(N-1), \overline{\varepsilon}_n^f(N-1))$	(5)	2 0 1 0
Total: $9n_{max}$ *înmulţiri;* 1 *împărţire;* $5n_{max}$ *adunări;* $2n_{max}$ *radicali*		

Tabelul 8.1. Algoritm latice pentru erorile de predicţie normalizate.

Scanned with OKEN Scanner

Ecuaţiile recursive (8.28), (8.31) şi (8.34) formează cea mai frecvent utilizată secţiune latice a erorilor normalizate. Iniţializarea pentru ordinul fictiv $n = 0$ se va face cu valorile care rezultă din ecuaţiile de iniţializare a secţiunii latice a Algoritmului 7.3.1., aplicând definiţiile (8.26) şi (8.27):

$$\overline{\varepsilon}_0^f(N) = (\sigma_0(N))^{-1/2} y(N), \tag{8.47}$$

$$\overline{\varepsilon}_0^b(N) = (\sigma_0(N-1))^{-1/2} y(N-1), \tag{8.48}$$

fiind necesară deci propagarea în timp şi a dispersiei $\sigma_0(N)$, ca în Algoritmul 7.3.1.

În Tabelul 8.1. este prezentat algoritmul latice pentru erorile normalizate de predicţie. În [28] se demonstrează că iniţializarea la pasul $N = 0$ a mărimilor $\overline{\alpha}_n(0)$ şi $\rho_0(0)$ cu valori nule şi apoi iterarea ecuaţiilor secţiunii latice, cu convenţia de a se atribui valoarea zero rezultatului operaţiilor de împărţire la care împărţitorul este zero, conduce la o iniţializare exactă a algoritmului latice normalizată.

Algoritmul 8.1.1. reprezintă doar o modalitate foarte atractivă din punct de vedere numeric de propagare în timp a mărimilor care descriu seria de timp ca proces AR de ordin $n \leq n_{max}$. Deşi nu este utilizat ca algoritm independent, el este inclus de obicei în algoritmi mai complecşi (cum ar fi cel pentru determinarea predictorilor) sub forma unei secţiuni iterative în timp.

Pentru aplicaţiile de predicţie, prezintă un mare interes transformarea inversă, de la mărimile normalizate la cele nenormalizate, care va fi dedusă din relaţiile (8.19), (8.20), (8.24) şi (8.30). În Tabelul 8.2. este prezentată secţiunea recursivă în ordin, care poate fi executată numai la momentele de timp dorite, pentru calculul erorilor de predicţie nenormalizate, plecând numai de la informaţia propagată în timp în Algoritmul 8.1.1.

8.2. Algoritmul latice pentru predictorii normalizaţi

Problema determinării predictorilor (Problema 7.3.1.) se poate rezolva şi plecând de la mărimile din algoritmul latice normalizată (Tabelul 8.1.) astfel: la momentele de timp în care se doreşte evaluarea predictorilor, se va executa pasul 2.3 din Tabelul 8.2. (realizând trecerea la mărimile nenormalizate) şi apoi se va executa pasul 2.3 din Tabelul 7.8. (pentru trecerea de la laticea nenormalizată la predictori). Efortul de calcul necesar în această schemă ar fi de ordinul $\mathcal{O}(n_{max}^2)$.

O soluţie alternativă, care va fi prezentată în continuare, o constituie modificarea prin normalizare a laticii predictorilor. Această variantă este avantajoasă în implementările în virgulă fixă.

Definim predictorii normalizaţi:

$$\overline{\theta}_n(N) = \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} \frac{1}{\sqrt{\sigma_n(N)}}, \tag{8.49}$$

$$\overline{\theta}_n^b(N) = \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \frac{1}{\sqrt{\sigma_n^b(N)}}, \tag{8.50}$$

$$\overline{K}_n^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} \frac{1}{\sqrt{\gamma_n^0(N)}}, \tag{8.51}$$

obţinuţi prin extinderea dimensiunii predictorilor şi normarea în raport cu "deviaţia standard de eşantion".

Extinderea Algoritmului 8.1.1. pentru calculul erorilor de predicţie nenormalizate

SECŢIUNEA OPŢIONALĂ PENTRU CALCULUL ERORILOR DE PREDICŢIE NENORMALIZATE		Num. op. ⋆ / + √
2. Recursii în timp		
2.3 Dacă la pasul N se doreşte evaluarea erorilor de predicţie nenormalizate Atunci:		
2.3.1 Iniţializare recursii în ordin:		
$\rho_0(N) = \sigma_0(N)$		
$\sigma_0^b(N) = \rho_0(N-1) = \sigma_0(N-1)$		
$\gamma_0^0(N) = 1$		
2.3.2 Recursii în ordin:		
Pentru $n = 0, \ldots, n_{max} - 1$		
$\sigma_{n+1}(N) = \sigma_n(N)(1 - \overline{\alpha}_n^2(N))$	(6)	2 0 1 0
$\sigma_n^b(N-1) = \sigma_n^b(N)(1 - \overline{\varepsilon}_n^{b2}(N))$	(7)	2 0 1 0
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1)(1 - \overline{\alpha}_n^2(N-1))$	(8)	2 0 1 0
$\gamma_{n+1}^0(N) = \gamma_n^0(N)(1 - \overline{\varepsilon}_n^{b2}(N))$	(9)	2 0 1 0
$\varepsilon_{n+1}^f(N) = \overline{\varepsilon}_{n+1}^f(N)\sqrt{\sigma_{n+1}(N)\gamma_{n+1}^0(N)}$	(10)	2 0 0 1
$\varepsilon_{n+1}^b(N) = \overline{\varepsilon}_{n+1}^b(N)\sqrt{\sigma_{n+1}^b(N)\gamma_{n+1}^0(N)}$	(11)	2 0 0 1
$\alpha_n(N) = \overline{\alpha}_n(N)\sqrt{\sigma_n(N)\sigma_n^b(N)}$	(12)	2 0 0 1
Total: $14n_{max}$ înmulţiri; 0 împărţiri; $4n_{max}$ adunări; $3n_{max}$ radicali		

Tabelul 8.2. Extinderea Algoritmului 8.1.1 cu o secţiune pentru trecerea de la erorile de predicţie normalizate la cele nenormalizate.

Scanned with OKEN Scanner

Lema 8.2.1. *Între predictorii normalizaţi (8.49)-(8.51) există următoarele relaţii recursive în ordin:*

$$\overline{\theta}_{n+1}(N) = (1-\overline{\alpha}_n^2(N))^{-1/2}\left\{\begin{bmatrix}\overline{\theta}_n(N)\\0\end{bmatrix} - \overline{\alpha}_n(N)\begin{bmatrix}0\\\overline{\theta}_n^b(N)\end{bmatrix}\right\}, \tag{8.52}$$

$$\overleftarrow{\theta}_{n+1}^b(N) = (1-\overline{\alpha}_n^2(N-1))^{-1/2}\left\{\begin{bmatrix}0\\\overleftarrow{\theta}_n^b(N-1)\end{bmatrix} - \overline{\alpha}_n(N-1)\begin{bmatrix}\overline{\theta}_n(N-1)\\0\end{bmatrix}\right\}, \tag{8.53}$$

$$\overline{K}_{n+1}^*(N) = (1-\overline{\varepsilon}_n^{b2}(N))^{-1/2}\left\{\begin{bmatrix}\overline{K}_n^*(N)\\0\end{bmatrix} - \overline{\varepsilon}_n^b(N)\begin{bmatrix}\overline{\theta}_n^b(N)\\0\end{bmatrix}\right\}. \tag{8.54}$$

Demonstraţie. Din relaţiile (8.49) şi (7.123) rezultă

$$\sqrt{\sigma_{n+1}(N)}\overline{\theta}_{n+1}(N) = \begin{bmatrix}1\\\theta_{n+1}(N)\end{bmatrix} = \begin{bmatrix}1\\\theta_n(N)\\0\end{bmatrix} - \frac{\alpha_n(N)}{\sigma_n^b(N)}\begin{bmatrix}0\\\theta_n^b(N)\\1\end{bmatrix} =$$

$$= \sqrt{\sigma_n(N)}\left\{\begin{bmatrix}\overline{\theta}_n(N)\\0\end{bmatrix} - \frac{\alpha_n(N)}{\sqrt{\sigma_n(N)}\sigma_n^b(N)}\begin{bmatrix}0\\\overleftarrow{\theta}_n^b(N)\end{bmatrix}\sqrt{\sigma_n^b(N)}\right\} \tag{8.55}$$

şi utilizând (8.19) rezultă (8.52).

Similar, din relaţiile (8.50) şi (7.124) rezultă

$$\sqrt{\sigma_{n+1}^b(N)}\overleftarrow{\theta}_{n+1}^b(N) = \begin{bmatrix}\theta_{n+1}^b(N)\\1\end{bmatrix} = \begin{bmatrix}0\\\theta_n^b(N-1)\\1\end{bmatrix} - \frac{\alpha_n(N-1)}{\sigma_n(N-1)}\begin{bmatrix}1\\\theta_n(N-1)\\0\end{bmatrix} = \tag{8.56}$$

$$= \sqrt{\sigma_n^b(N-1)}\left\{\begin{bmatrix}0\\\overleftarrow{\theta}_n^b(N-1)\end{bmatrix} - \frac{\alpha_n(N-1)}{\sqrt{\sigma_n^b(N-1)}\sigma_n(N-1)}\begin{bmatrix}\overline{\theta}_n(N-1)\\0\end{bmatrix}\sqrt{\sigma_n(N-1)}\right\}$$

şi utilizând (8.20) rezultă (8.53).

Din (8.51) şi (7.131) rezultă

$$\sqrt{\gamma_{n+1}^0(N)}\overline{K}_{n+1}^*(N) = \begin{bmatrix}K_{n+1}^*(N)\\0\end{bmatrix} = \begin{bmatrix}K_n^*(N)\\0\\0\end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)}\begin{bmatrix}\theta_n^b(N)\\1\\0\end{bmatrix} = \tag{8.57}$$

$$= \sqrt{\gamma_n^0(N)}\left\{\begin{bmatrix}\overline{K}_n^*(N)\\0\end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sqrt{\gamma_n^0(N)}\sigma_n^b(N)}\begin{bmatrix}\overleftarrow{\theta}_n^b(N)\\0\end{bmatrix}\sqrt{\sigma_n^b(N)}\right\}$$

şi utilizând (8.30) rezultă (8.54).

□

Pentru a "decupla" recursiile în ordin de cele în timp în (8.53) sunt necesare relaţii recursive în timp pentru predictorii normalizaţi.

Lema 8.2.2. *Între predictorii normalizaţi* (8.49)-(8.51) *există următoarele relaţii recursive în timp*

$$\begin{bmatrix}\overline{\theta}_n(N-1)\\0\end{bmatrix} = (1-\overline{\varepsilon}_n^{f2}(N))^{-1/2}\left\{\begin{bmatrix}\overline{\theta}_n(N)\\0\end{bmatrix} - \overline{\varepsilon}_n^f(N)\begin{bmatrix}0\\\overline{K}_n^*(N)\end{bmatrix}\right\}, \tag{8.58}$$

$$\overleftarrow{\theta}_n^b(N-1) = (1-\overline{\varepsilon}_n^{b2}(N))^{-1/2}(\overleftarrow{\theta}_n^b(N) - \overline{\varepsilon}_n^b(N)\overline{K}_n^*(N)). \tag{8.59}$$

Scanned with OKEN Scanner

Demonstraţie. Din ecuaţiile (7.94),(7.99) şi (7.100) rezultă

$$\theta_n(N-1) = \theta_n(N) - K_n^*(N)\frac{\varepsilon_n^f(N)}{\gamma_n^0(N)}, \tag{8.60}$$

sau, utilizând definiţiile mărimilor normalizate,

$$\sqrt{\sigma_n(N-1)}\begin{bmatrix}\overline{\theta}_n(N-1)\\0\end{bmatrix} = \begin{bmatrix}1\\\theta_n(N-1)\\0\end{bmatrix} = \begin{bmatrix}1\\\theta_n(N)\\0\end{bmatrix} - \frac{\varepsilon_n^f(N)}{\gamma_n^0(N)}\begin{bmatrix}0\\K_n^*(N)\\0\end{bmatrix} =$$

$$= \sqrt{\sigma_n(N)}\left\{\begin{bmatrix}\overline{\theta}_n(N)\\0\end{bmatrix} - \frac{\varepsilon_n^f(N)}{\sqrt{\sigma_n(N)}\gamma_n^0(N)}\begin{bmatrix}0\\\overline{K}_n^*(N)\end{bmatrix}\sqrt{\gamma_n^0(N)}\right\} \tag{8.61}$$

şi ţinând cont de (8.23) rezultă (8.58).

Din ecuaţiile (7.95), (7.99) şi (7.101) rezultă

$$\theta_n^b(N-1) = \theta_n^b(N) - K_n^*(N)\frac{\varepsilon_n^b(N)}{\gamma_n^0(N)}, \tag{8.62}$$

sau, introducând definiţiile predictorilor normalizaţi,

$$\sqrt{\sigma_n^b(N-1)}\overline{\theta}_n^b(N-1) = \begin{bmatrix}\theta_n^b(N)\\1\end{bmatrix} - \frac{\varepsilon_n^b(N)}{\gamma_n^0(N)}\begin{bmatrix}K_n^*(N)\\0\end{bmatrix} =$$

$$= \sqrt{\sigma_n^b(N)}\left\{\overline{\theta}_n^b(N) - \frac{\varepsilon_n^b(N)}{\sqrt{\sigma_n^b(N)}\gamma_n^0(N)}\overline{K}_n^*(N)\sqrt{\gamma_n^0(N)}\right\} \tag{8.63}$$

şi utilizând (8.22) rezultă (8.59).

Ecuaţiile (8.52)–(8.54) împreună cu (8.58) şi (8.59) formează un set complet de recursii, furnizând valorile predictorilor normalizaţi la un moment oarecare N.

Este de aşteptat ca în cele mai multe aplicaţii să fie necesar predictorul înainte nenormalizat, care se poate obţine plecând de la predictorul normalizat, utilizând relaţia (8.49), în modul următor:

$$(\overline{\theta}_n(N))_1 = (\sigma_n(N))^{-1/2}, \tag{8.64}$$

$$\begin{bmatrix}1\\\theta_n(N)\end{bmatrix} = (\overline{\theta}_n(N))_1^{-1}\overline{\theta}_n(N). \tag{8.65}$$

Iniţializarea algoritmului pentru ordinul fictiv $n = 0$ rezultă imediat din relaţiile de definiţie (8.49)-(8.51):

$$\overline{\theta}_0(N) = (\sigma_0(N))^{-1/2};\ \overline{\theta}_0^b(N) = (\sigma_0^b(N))^{-1/2};\ \overline{K}_0^*(N) = 0. \tag{8.66}$$

Să observăm că toate ecuaţiile implicate în calculul recursiv al predictorilor normalizaţi reprezintă transformări prin funcţia

$$G(a,b,c) = (1-c^2)^{-1/2}(a-cb), \tag{8.67}$$

ceea ce conduce la un algoritm cu o structură foarte ordonată, convenabilă în implementările pe structuri de calcul paralele. Acest algoritm este prezentat în Tabelul 8.3.

Scanned with OKEN Scanner

Extinderea Algoritmului 8.1.1. pentru calculul predictorilor

SECŢIUNEA OPŢIONALĂ PENTRU CALCULUL PREDICTORILOR

2. Recursii în timp

2.4 Dacă la pasul N se doreşte evaluarea predictorilor nenormalizaţi

Atunci:

2.4.1 Iniţializare recursii în ordin:

$$\overline{\theta}_0(N) = (\sigma_0(N))^{-1/2}$$

$$\overline{\theta}_0^b(N) = (\sigma_0^b(N))^{-1/2}$$

$$\overline{K_0^*}(N) = 0$$

2.4.2 Recursii în ordin:

Pentru $n = 0, \ldots, n_{max} - 1$

$$\overline{\theta}_{n+1}(N) = G\left(\begin{bmatrix} \overline{\theta}_n(N) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \overline{\theta}_n^b(N-1) \end{bmatrix}, \overline{\alpha}_n(N)\right)$$

$$\begin{bmatrix} \overline{\theta}_n(N-1) \\ 0 \end{bmatrix} = G\left(\begin{bmatrix} \overline{\theta}_n(N) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \overline{K_n^*}(N) \end{bmatrix}, \overline{\varepsilon}_n^f(N)\right)$$

$$\overline{\theta}_n^b(N-1) = G(\overline{\theta}_n^b(N), \overline{K_n^*}(N), \overline{\varepsilon}_n^b(N))$$

$$\overline{\theta}_{n+1}^b(N) = G\left(\begin{bmatrix} 0 \\ \overline{\theta}_n^b(N-1) \end{bmatrix}, \begin{bmatrix} \overline{\theta}_n(N-1) \\ 0 \end{bmatrix}, \overline{\alpha}_n(N-1)\right)$$

$$\overline{K_{n+1}^*}(N) = G\left(\begin{bmatrix} \overline{K_n^*}(N) \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{\theta}_n^b(N) \\ 0 \end{bmatrix}, \overline{\varepsilon}_n^b(N)\right)$$

$$\begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} = [\overline{\theta}_n(N)]_1^{-1} \overline{\theta}_n(N)$$

Tabelul 8.3. Extinderea Algoritmului 8.1.1 cu o secţiune pentru trecerea de la erorile de predicţie normalizate la predictorii normalizaţi şi nenormalizaţi.

Scanned with OKEN Scanner

8.3. Parametrizarea latice normalizată a modelelor AR

Informaţia disponibilă în secţiunea latice din Algoritmul 8.1.1. este suficientă pentru deducerea atât a erorilor de predicţie nenormalizate, cât şi a predictorilor. În consecinţă, laticea normalizată poate fi considerată o altă parametrizare a modelelor AR optimale de ordine între 1 şi n_{max}.

Datorită faptului că algoritmul latice normalizată este o implementare exactă a metodei CMMP cu criteriul determinist, parametrii predictorilor înainte şi înapoi (deci a modelelor AR) vor converge în condiţii destul de generale [76]. Vom considera în continuare problema relaţiei dintre predictori (vom presupune că valorile predictorilor furnizate de algoritm au convers la valorile optime, şi vom suprima dependenţa acestora de timpul N) şi parametrii latice normalizaţi $\overline{\alpha}_n$.

În Figura 8.1.*a* este reprezentată structura sistemului discret care realizează prelucrările descrise de ecuaţiile (8.11)–(8.14) (s-a notat $k_i = \overline{\alpha}_i(N)$). Structura unui asemenea filtru se numeşte latice normalizată.

Pentru a evidenţia proprietăţile de transfer între intrarea $y(t)$ şi ieşirile $\tilde{\varepsilon}_i^f(t)$ şi $\tilde{\varepsilon}_i^b(t)$ ale filtrului latice normalizată (caracterizat prin parametrii invariabili în timp k_i), vom aplica la intrarea filtrului în Figura 8.1.*b* un impuls unitar $\delta(t)$ ($\mathcal{Z}\{\delta(t)\} = 1$). Vom nota, ca în figură, funcţiile de transfer între intrare şi ieşirile $\tilde{\varepsilon}_i^f(t)$ prin $A_i(z^{-1})$, iar funcţiile de transfer între intrare şi ieşirile $\tilde{\varepsilon}_i^b(t+1)$ prin $B_i(z^{-1})$. Din Figura 8.1.*b* rezultă următoarele relaţii recursive între funcţiile de transfer astfel definite:

$$\overline{A}_{n+1}(z^{-1}) = (1-k_n^2)^{-1/2}\left(\overline{A}_n(z^{-1}) - k_n z^{-1}\overline{B}_n(z^{-1})\right), \tag{8.68}$$

$$\overline{B}_{n+1}(z^{-1}) = (1-k_n^2)^{-1/2}\left(z^{-1}\overline{B}_n(z^{-1}) - k_n\overline{A}_n(z^{-1})\right). \tag{8.69}$$

Dacă se doreşte trecerea la funcţia de transfer nenormalizată $A_n(z^{-1})$ definită de

$$\varepsilon_n^f(t) = A_n(q^{-1})y(t) = (1 + a_1 q^{-1} + \ldots + a_n q^{-n})y(t), \tag{8.70}$$

prin repetarea evaluării relaţiei (8.68) în $z^{-1} = 0$ şi utilizând (8.1), rezultă

$$A_n(z^{-1}) = \overline{A}_n(z^{-1})/\overline{A}_n(0) = \overline{A}_n(z^{-1})\prod_{i=1}^{n}(1-k_n^2)^{-1/2}\sqrt{\sigma_0}. \tag{8.71}$$

În concluzie, în cazul în care parametrii $\overline{\alpha}_i(N)$ nu depind de argumentul timp N, relaţiile de trecere de la parametrizarea latice normalizată la parametrii AR sunt date de (8.68),(8.69) şi (8.71).

În cazul general, trecerea de la prametrii $\overline{\alpha}_i(N)$ la parametrii AR se realizează prin secţiunea predictorilor din Tabelul 8.3.

Să definim funcţiile de transfer:

$$\overline{A}_{n,N}(z^{-1}) = \begin{bmatrix} 1 & z^{-1} & \ldots & z^{-n} \end{bmatrix}\overline{\theta}_n(N), \tag{8.72}$$

$$\overline{B}_{n,N}(z^{-1}) = \begin{bmatrix} 1 & z^{-1} & \ldots & z^{-n} \end{bmatrix}\overline{\theta}_n^b(N) \tag{8.73}$$

şi să notăm:

$$\tilde{\varepsilon}_{n,N}^f(t) \triangleq \overline{A}_{n,N}(q^{-1})y(t) = (y(t) - \varphi_n^{\mathrm{T}}(t)\theta_n(N))/\sqrt{\sigma_n(N)}, \tag{8.74}$$

$$\tilde{\varepsilon}_{n,N}^b(t+1) \triangleq \overline{B}_{n,N}(q^{-1})y(t) = (y(t-n) - \varphi_n^{\mathrm{T}}(t+1)\theta_n^b(N))/\sqrt{\sigma_n^b(N)}. \tag{8.75}$$

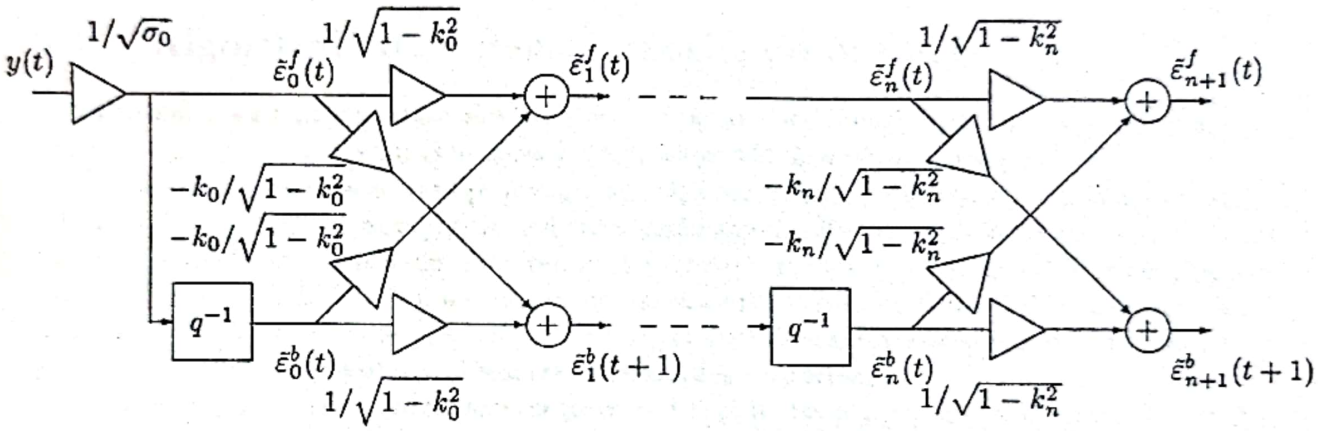
Scanned with OKEN Scanner

Predictorii normalizaţi calculaţi prin ecuaţiile din Tabelul 8.3 caracterizează transferul de la intrarea $y(t)$ la ieşirile (8.74) şi (8.75). Funcţiile de transfer definite de (8.72), (8.73) satisfac următoarele relaţii recursive obţinute prin înmulţirea ecuaţiilor (8.52) şi (8.53) la stânga cu $\left[\begin{array}{cccc} 1 & z^{-1} & \ldots & z^{-n-1} \end{array}\right]$:

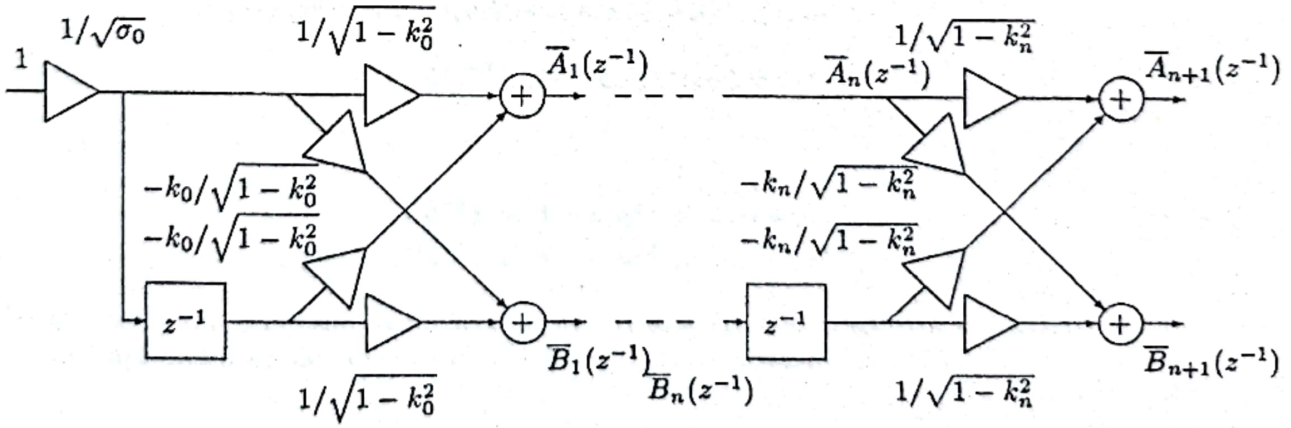
$$\overline{A}_{n+1,N}(z^{-1}) = (1-\overline{\alpha}_n^2(N))^{-1/2}\left\{\overline{A}_{n,N}(z^{-1}) - \overline{\alpha}_n(N)z^{-1}\overline{B}_{n,N}(z^{-1})\right\}, \quad (8.76)$$

$$\overline{B}_{n+1,N+1}(z^{-1}) = (1-\overline{\alpha}_n^2(N))^{-1/2}\left\{z^{-1}\overline{B}_{n,N}(z^{-1}) - \overline{\alpha}_n(N)\overline{A}_{n,N}(z^{-1})\right\}, \quad (8.77)$$

relaţii similare cu (8.68), (8.69) care definesc laticea normalizată. Se observă că atât pentru calculul rapid al erorilor de predicţie, cât şi pentru calculul recursiv al predictorilor, intervine structura latice normalizată. Această structură are proprietăţi numerice superioare structurii latice nenormalizate şi este preferată în structurile specializate în care pot fi utilizate circuitele de tip CORDIC pentru efectuarea operaţiilor care definesc funcţia G [34].



a) Structura latice normalizată pentru erorile de predicţie



b) Funcţiile de transfer ale etajelor structurii latice

Figura 8.1. Structura latice normalizată: *(a)* prelucrarea semnalului de intrare $y(t)$; *(b)* funcţiile de transfer ale etajelor structurii latice.

Scanned with OKEN Scanner

Capitolul 9

Algoritmul Latice pentru Modele ARMA Monovariabile

9.1. Algoritmi latice pentru erorile de predicţie

În Capitolul 3 au fost introduse modelele ARMA şi au fost deduşi algoritmi rapizi recursivi în ordin de tip Levinson pentru estimarea parametrilor AR ai acestor modele.

În acest capitol va fi rezolvată problema estimării recursive, în ordin şi în timp, a parametrilor AR ai modelelor ARMA(n, m), prin utilizarea estimatorilor de variabilă instrumentală (VI). Se va presupune că segmentul de date poate fi prelungit pentru $t < 0$ cu valori nule (ipoteza "fereastră anterioară"). Se poate arăta că estimatorul dedus în Capitolul 3 aparţine clasei metodelor VI şi corespunde presupunerii că segmentul de date a fost prelungit cu valori nule pentru $t < 0$ şi $t > N$- ipoteza "fereastră anterioară şi posterioară".

Vor fi deduse relaţii pentru recurenţa în ordin şi în timp a estimaţiilor parametrilor AR ai modelelor ARMA şi a mărimilor auxiliare necesare. În final, prin selectarea dintre relaţiile recursive a unor seturi complete de ecuaţii vor rezulta algoritmi latice pentru erorile de predicţie iar în Secţiunea 9.2 va fi dedus algoritmul pentru trecerea de la parametrizarea latice a secţiunii AR la parametrizarea directă.

Fie semnalul $y(t)$ generat de următorul model ARMA(n, m)

$$A(q^{-1})y(t) = C(q^{-1})e(t) = \nu(t), \tag{9.1}$$

în care

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}, \tag{9.2}$$
$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_m q^{-m}, \tag{9.3}$$

iar $e(t)$ este un zgomot alb de medie zero şi $\nu(t)$ este zgomot corelat (proces MA).

În Capitolul 3 s-a demonstrat că parametrii AR ai modelului ARMA,

$$\theta_n = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}^{\mathrm{T}}, \tag{9.4}$$

pot fi obţinuţi ca soluţii ale sistemului de ecuaţii

$$R_n \theta_n = -r_n, \tag{9.5}$$

în care:

$$R_n = \begin{bmatrix} \rho_m & \rho_{m-1} & \cdots & \rho_{m+1-n} \\ \rho_{m+1} & \rho_m & \cdots & \rho_{m+2-n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \rho_{m+n-1} & \rho_{m+n-2} & \cdots & \rho_m \end{bmatrix}, \tag{9.6}$$

$$r_n = \begin{bmatrix} \rho_{m+1} \\ \cdot \\ \cdot \\ \cdot \\ \rho_{m+n} \end{bmatrix}, \tag{9.7}$$

$$\rho_s = Ey(t)y(t-s). \tag{9.8}$$

În Capitolul 3 au fost obţinute estimaţii ale parametrilor AR înlocuind în (9.5)–(9.7) valorile funcţiei de autocorelaţie (9.8) prin estimaţii ale acesteia, de exemplu

$$\rho_s(N) = \frac{1}{N}\sum_{t=0}^{N} y(t)y(t-s). \tag{9.9}$$

Să observăm că mărimile (9.6),(9.7) se pot rescrie, dacă utilizăm notaţiile:

$$z(t) = y(t-m), \tag{9.10}$$

$$\varphi_n(t) = \left[\, y(t-1) \;\ldots\; y(t-n) \,\right]^{\mathrm{T}}, \tag{9.11}$$

$$z_n(t) = \left[\, z(t-1) \;\ldots\; z(t-n) \,\right]^{\mathrm{T}}, \tag{9.12}$$

în modul următor

$$R_n = E \begin{bmatrix} y(t-m-1) \\ y(t-m-2) \\ \cdot \\ \cdot \\ \cdot \\ y(t-m-n) \end{bmatrix} \left[\, y(t-1) \;\ldots\; y(t-n) \,\right] =$$

$$= E \begin{bmatrix} z(t-1) \\ z(t-2) \\ \cdot \\ \cdot \\ \cdot \\ z(t-n) \end{bmatrix} \left[\, y(t-1) \;\ldots\; y(t-n) \,\right] = Ez_n(t)\varphi_n^{\mathrm{T}}(t), \tag{9.13}$$

$$r_n = E \begin{bmatrix} y(t-m-1) \\ y(t-m-2) \\ \cdot \\ \cdot \\ \cdot \\ y(t-m-n) \end{bmatrix} y(t) = Ez_n(t)y(t). \tag{9.14}$$

Considerând că valorile seriei $y(t)$ pentru $t < 0$ sunt nule, din (9.13) şi (9.14) rezultă următoarele estimaţii scalate prin N ale mărimilor R_n şi r_n:

$$R_n(N) = \sum_{t=0}^{N} z_n(t)\varphi_n^{\mathrm{T}}(t), \tag{9.15}$$

$$r_n(N) = \sum_{t=0}^{N} z_n(t)y(t). \tag{9.16}$$

În consecinţă, o altă estimaţie, $\theta_n(N)$, a vectorului parametrilor va fi definită ca fiind soluţia sistemului

$$R_n(N)\theta_n(N) = -r_n(N) \tag{9.17}$$

şi este de fapt o estimaţie de variabliă instrumentală, corespunzătoare vectorului instrumentelor (9.12). Impunând condiţia ca vectorul $z_n(t)$ să satisfacă relaţia

$$Ez_n(t)\nu(t) = Ez_n(t)(y(t) + \varphi_n^{\mathrm{T}}(t)\theta) = 0, \tag{9.18}$$

care este necesară pentru consistenţa estimaţiei[76], va rezulta (9.5) (aceasta fiind o altă justificare a utilizării sistemului (9.5)). Estimatorul (9.5)-(9.7), în care funcţiile de covarianţă (9.8) se înlocuiesc cu estimaţiile acestora (9.9), este un caz particular al estimatorului (9.17): se constată imediat că dacă presupunem că seria $y(t)$ este nulă în afara segmentului $0 \le t \le N$, atunci elementele matricelor $R_n(N+n)$ şi vectorului $r_n(N+n)$ sunt autocovarianţele de eşantion (9.9), deci $\theta_n(N+n)$ este identic cu θ_n.

Problema 9.1.1. Fiind date valorile seriei $\{y(t)\}_{t=0}^{N}$ şi făcând următoarele presupuneri:

$$y(t) = 0 \text{ pentru } t < 0, \tag{9.19}$$

$$\text{matricele } R_n(N) \text{ sunt nesingulare pentru } n = 1, \ldots, n_{max}, \tag{9.20}$$

să se găsească algoritmi eficienţi numeric pentru estimarea prin metoda VI a parametrilor latice ai secţiunii AR a modelelor ARMA, corespunzători soluţiilor $\{\theta_n\}_{n=1}^{n_{max}}$ ale sistemelor (9.17).

Observaţii 1. Eficienţa numerică a algoritmilor latice VI va fi asemănătoare cu cea a algoritmilor latice de tip CMMP, datorită similitudinii sistemelor (9.17) şi (7.38) (principala deosebire fiind faptul că matricea $R_n(N)$ din (9.15) nu este simetrică pe când matricea din (7.34) este simetrică). Comparând soluţionarea clasică a celor n_{max} sisteme (9.17) pentru a obţine modelele $\{\theta_n\}_{n=1}^{n_{max}}$, necestând $\mathcal{O}(n_{max}^4)$ operaţii, algoritmii latice VI vor reduce numărul de operaţii necesar la $\mathcal{O}(n_{max})$ pentru determinarea parametrilor latice şi la $\mathcal{O}(n_{max}^2)$ pentru trecerea la parametrizarea directă $\{\theta_n\}_{n=1}^{n_{max}}$ (realizată prin algoritmii din Secţiunea 9.2).

2. Există algoritmi latice VI pentru care ipoteza (9.19) nu este necesară. Complexitatea acestora este însă aproximativ de două ori mai mare decât a celor prezentaţi în continuare, fără ca aceasta să fie compensată de o creştere corespunzătoare a performanţelor algoritmilor decât pentru lungimi N mici ale segmentului de date. Perturbaţia introdusă prin anularea unora din valorile iniţiale ale seriei (conform ipotezei (9.19)) este cu atât mai bine compensată cu cât un număr mai mare N de date valide sunt considerate.

Scanned with OKEN Scanner

Relaţii recursive în ordin

Partiţiile vectorilor $\varphi_{n+1}(t)$ şi $z_{n+1}(t)$:

$$\varphi_{n+1}(t) = \begin{bmatrix} \varphi_n(t) \\ y(t-n-1) \end{bmatrix} = \tag{9.21}$$

$$= \begin{bmatrix} y(t-1) \\ \varphi_n(t-1) \end{bmatrix}, \tag{9.22}$$

$$z_{n+1}(t) = \begin{bmatrix} z_n(t) \\ z(t-n-1) \end{bmatrix} = \tag{9.23}$$

$$= \begin{bmatrix} z(t-1) \\ z_n(t-1) \end{bmatrix} \tag{9.24}$$

generează partiţii pentru toate mărimile implicate în calculul estimaţiei $\theta_n(N)$:

$$R_{n+1}(N) = \sum_{t=0}^{N} z_{n+1}(t)\varphi_{n+1}(t) = \begin{bmatrix} R_n(N) & r_n^b(N) \\ \overline{r}_n^{bT}(N) & \overline{\rho}_0(N-n-1) \end{bmatrix} = \tag{9.25}$$

$$= \begin{bmatrix} \overline{\rho}_0(N-1) & \overline{r}_n^T(N-1) \\ r_n(N-1) & R_n(N-1) \end{bmatrix} \tag{9.26}$$

şi

$$r_{n+1}(N) = \sum_{t=0}^{N} z_{n+1}(t)y(t) = \begin{bmatrix} r_n(N) \\ \overline{\rho}_{n+1}(N) \end{bmatrix}. \tag{9.27}$$

Datorită numărului mare de mărimi auxiliare care vor interveni în continuare, pentru facilitarea referirii şi o mai mare claritate, definiţiile tuturor mărimilor implicate în algoritmii din acest capitol sunt grupate în Tabelul 9.1.

Este de notat similitudinea notaţiilor adoptate cu notaţiile utilizate în Capitolul 7, dar trebuie remarcate şi următoarele particularităţi:

- au fost suprabarate mărimile în a căror definire intervine semnalul scalar $z(\cdot)$, sau vectorul $\varphi_N(t)$, sau o altă mărime suprabarată;
- mărimile auxiliare introduse pentru a facilita obţinerea de algoritmi rapizi nu mai au semnificaţii care să le facă utile "per se" (predictori optimali, dispersii optimale, erori de predicţie optimale), cum a fost cazul la algoritmii rapizi CMMP, de aceea denumirile au fost transferate între cele două categorii de algoritmi numai datorită similitudinii formale de notaţii utilizate - de unde şi sublinierea caracterului "figurat" al denumirilor prin ghilimele.

Pentru vectorii de intercorelaţie auxiliari (9.29)–(9.31), partiţiile (9.21)–(9.24) generează următoarele relaţii recursive în ordin:

$$r_{n+1}^b(N) = \sum_{t=0}^{N} z_{n+1}(t)y(t-n-2) = \begin{bmatrix} \overline{\rho}_{-n-1}(N-n-2) \\ r_n^b(N-1) \end{bmatrix}, \tag{9.55}$$

$$\overline{r}_{n+1}(N) = \sum_{t=0}^{N} \varphi_{n+1}(t)z(t) = \begin{bmatrix} \overline{r}_n(N) \\ \overline{\rho}_{-n-1}(N-n-1) \end{bmatrix}, \tag{9.56}$$

$$\overline{r}_{n+1}^b(N) = \sum_{t=0}^{N} \varphi_{n+1}(t)z(t-n-2) = \begin{bmatrix} \overline{\rho}_{n+1}(N-1) \\ \overline{r}_n^b(N-1) \end{bmatrix}. \tag{9.57}$$

Scanned with OKEN Scanner

Vectori de intercorelaţie

$$r_n(N) = \sum_{t=0}^{N} z_n(t)y(t) = \sum_{t=0}^{N} \varphi_n(t-m)y(t) \tag{9.28}$$

$$r_n^b(N) = \sum_{t=0}^{N} z_n(t)y(t-n-1) \tag{9.29}$$

$$\overline{r}_n(N) = \sum_{t=0}^{N} \varphi_n(t)z(t) \tag{9.30}$$

$$\overline{r}_n^b(N) = \sum_{t=0}^{N} \varphi_n(t)z(t-n-1) \tag{9.31}$$

Funcţii de intercorelaţie

$$\overline{\rho}_0(N) = \sum_{t=0}^{N} z(t)y(t) = \rho_m(N) \tag{9.32}$$

$$\overline{\rho}_p(N) = \sum_{t=0}^{N} z(t-p)y(t) = \rho_{m+p}(N) \tag{9.33}$$

$$\overline{\rho}_{n+1}(N) = \sum_{t=0}^{N} z(t-n-1)y(t) = \rho_{m+n+1}(N) \tag{9.34}$$

$$\overline{\rho}_{-n-1}(N-n-1) = \sum_{t=0}^{N} z(t)y(t-n-1) = \rho_{m-n-1}(N-n-1) \tag{9.35}$$

Predictori

$$\theta_n(N) = -R_n^{-1}(N)r_n(N) \tag{9.36}$$

$$\theta_n^b(N) = -R_n^{-1}(N)r_n^b(N) \tag{9.37}$$

$$\overline{\theta}_n(N) = -R_n^{-T}(N)r_n(N) \tag{9.38}$$

$$\overline{\theta}_n^b(N) = -R_n^{-T}(N)r_n^b(N) \tag{9.39}$$

$$K_n^*(N) = -R_n^{-1}(N)z_n(N) \tag{9.40}$$

$$\overline{K}_n^*(N) = -R_n^{-T}(N)\varphi_n(N) \tag{9.41}$$

Dispersii

$$\sigma_n(N) = \overline{\rho}_0(N) - \overline{r}_n^T(N)R_n^{-1}(N)r_n(N) = \overline{\rho}_0(N) - \overline{r}_n^T(N)\theta_n(N) \tag{9.42}$$

$$\sigma_n^b(N) = \overline{\rho}_0(N-n-1) - \overline{r}_n^{bT}(N)R_n^{-1}(N)r_n^b(N) = \overline{\rho}_0(N-n-1) - \overline{r}_n^{bT}(N)\theta_n^b(N) \tag{9.43}$$

$$\alpha_n(N) = \overline{\rho}_{n+1}(N) - \overline{r}_n^{bT}(N)R_n^{-1}(N)r_n(N) \tag{9.44}$$

$$\overline{\alpha}_n(N) = \overline{\rho}_{-n-1}(N-n-1) - \overline{r}_n^T(N)R_n^{-1}(N)r_n^b(N) \tag{9.45}$$

Erori de predicţie

$$\varepsilon_n(N) = y(N) + \varphi_n^T(N)\theta_n(N) \tag{9.46}$$

$$\overline{\varepsilon}_n(N) = z(N) + z_n^T(N)\overline{\theta}_n(N) \tag{9.47}$$

$$e_n(N) = y(N) + \varphi_n^T(N)\theta_n(N-1) \tag{9.48}$$

$$\overline{e}_n(N) = z(N) + z_n^T(N)\overline{\theta}_n(N-1) \tag{9.49}$$

$$\varepsilon_n^b(N) = y(N-n-1) + \varphi_n^T(N)\theta_n^b(N) \tag{9.50}$$

$$\overline{\varepsilon}_n^b(N) = z(N-n-1) + z_n^T(N)\overline{\theta}_n^b(N) \tag{9.51}$$

$$\gamma_n^0(N) = 1 - \varphi_n^T(N)R_n^{-1}(N)z_n(N) \tag{9.52}$$

$$e_n^b(N) = y(N-n-1) + \varphi_n^T(N)\theta_n^b(N-1) \tag{9.53}$$

$$\overline{e}_n^b(N) = z(N-n-1) + z_n^T(N)\overline{\theta}_n^b(N-1) \tag{9.54}$$

Tabelul 9.1. Definiţii ale variabilelor utilizate în algoritmii latice de Variabilă Instrumentală.

Scanned with OKEN Scanner

Pentru a obţine relaţii recursive în ordin, pentru $\theta_n(N)$, este necesară stabilirea recurenţei în ordin pentru $R_n^{-1}(N)$. Aplicând lema inversării matricelor partiţionate (Lema 3.2.2.) şi utilizând partiţiile (9.25) şi (9.26) rezultă:

$$\begin{aligned} R_{n+1}^{-1}(N) &= \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -R_n^{-1}(N) r_n^b(N) \\ 1 \end{bmatrix} \left(\overline{\rho}_0(N-n-1) - \overline{r}_n^{bT}(N) R_n^{-1}(N) r_n^b(N) \right)^{-1} \cdot \\ &\quad \cdot \begin{bmatrix} -\overline{r}_n^{bT}(N) R_n^{-1}(N) & 1 \end{bmatrix} = \\ &= \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} (\sigma_n^b(N))^{-1} \begin{bmatrix} \overline{\theta}_n^{bT}(N) & 1 \end{bmatrix}, \end{aligned} \tag{9.58}$$

$$\begin{aligned} R_{n+1}^{-1}(N) &= \begin{bmatrix} 0 & 0 \\ 0 & R_n^{-1}(N-1) \end{bmatrix} + \begin{bmatrix} 1 \\ -R_n^{-1}(N-1) r_n(N-1) \end{bmatrix} \cdot \\ &\quad \cdot \left(\overline{\rho}_0(N-1) - \overline{r}_n^{T}(N-1) R_n^{-1}(N-1) r_n(N-1) \right)^{-1} \begin{bmatrix} 1 & -\overline{r}_n^{T}(N-1) R_n^{-1}(N-1) \end{bmatrix} = \\ &= \begin{bmatrix} 0 \\ I \end{bmatrix} R_n^{-1}(N-1) \begin{bmatrix} 0 & I \end{bmatrix} + \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} (\sigma_n(N-1))^{-1} \begin{bmatrix} 1 & \overline{\theta}_n^{T}(N-1) \end{bmatrix}. \end{aligned} \tag{9.59}$$

Relaţia recursivă în ordin pentru $\theta_{n+1}(N)$ se obţine imediat din partiţiile: (9.58) a matricei $R_{n+1}^{-1}(N)$ şi (9.27) a vectorului $r_{n+1}(N)$ şi utilizând definiţia (9.44), astfel

$$\theta_{n+1}(N) = -R_{n+1}^{-1}(N) r_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \frac{\alpha_n(N)}{\sigma_n^b(N)}. \tag{9.60}$$

În mod similar se obţin ecuaţii recursive pentru toţi predictorii. Aceste ecuaţii sunt grupate în Tabelul 9.3., iar modul lor de deducere este sintetizat în Tabelul 9.2.

Eroarea de predicţie $\varepsilon_{n+1}(N)$ se poate calcula utilizând partiţia (9.58) ca ecuaţie de bază şi înmulţind-o la stânga cu $\varphi_{n+1}^T(N)$ din (9.21) şi la dreapta cu $r_{n+1}(N)$ dat de (9.27):

$$\begin{aligned} \varepsilon_{n+1}(N) &= y(N) - \varphi_{n+1}^T(N) R_{n+1}^{-1}(N) r_{n+1}(N) = y(N) - \begin{bmatrix} \varphi_n^T(N) y(N-n-1) \end{bmatrix} \cdot \\ &\quad \cdot \left\{ \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \left(\sigma_n^b(N) \right)^{-1} \begin{bmatrix} \overline{\theta}_n^{bT}(N) & 1 \end{bmatrix} \right\} \begin{bmatrix} r_n(N) \\ \overline{\rho}_{n+1}(N) \end{bmatrix} = \\ &= \varepsilon_n(N) - \varepsilon_n^b(N) \frac{\alpha_n(N)}{\sigma_n^b(N)}, \end{aligned}$$

rezultând relaţia (9.67) din Tabelul 9.3. Pentru celelalte erori de predicţie ca şi pentru "dispersii" se procedează în mod similar prin înlocuirile sintetizate în Tabelul 9.2, rezultând ecuaţiile din Tabelul 9.3.

Relaţii recursive în timp

Recurenţa în timp a matricei $R_n(N)$ poate fi descrisă prin factorizarea

$$\begin{bmatrix} R_n(N) & z_n(N) \\ \varphi_n^T(N) & 1 \end{bmatrix} = \begin{bmatrix} I & z_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_n(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \varphi_n^T(N) & 1 \end{bmatrix}. \tag{9.75}$$

Scanned with OKEN Scanner

Ecuaţie de bază	Inmulţire la stânga	Inmulţire la dreapta	Ecuaţie rezultată
$R_{n+1}^{-1}(N)$ (9.58)		$r_{n+1}(N)$ (9.27)	(9.61)
$R_{n+1}^{-1}(N)$ (9.59)		$r_{n+1}^{b}(N)$ (9.55)	(9.62)
$R_{n+1}^{-1}(N)$ (9.58)		$\overline{r}_{n+1}(N)$ (9.56)	(9.63)
$R_{n+1}^{-1}(N)$ (9.59)		$\overline{r}_{n+1}^{b}(N)$ (9.57)	(9.64)
$R_{n+1}^{-1}(N)$ (9.58)		$\varphi_{n+1}(N)$ (9.21)	(9.65)
$R_{n+1}^{-1}(N)$ (9.59)		$\varphi_{n+1}(N)$ (9.22)	(9.66)
$R_{n+1}^{-1}(N)$ (9.58)	$\overline{r}_{n+1}^{T}(N)$ (9.56)	$r_{n+1}(N)$ (9.27)	(9.73)
$R_{n+1}^{-1}(N)$ (9.59)	$\overline{r}_{n+1}^{bT}(N)$ (9.57)	$r_{n+1}^{b}(N)$ (9.55)	(9.74)
$R_{n+1}^{-1}(N)$ (9.58)	$\varphi_{n+1}^{T}(N)$ (9.21)	$r_{n+1}(N)$ (9.27)	(9.67)
$R_{n+1}^{-1}(N)$ (9.59)	$\varphi_{n+1}^{T}(N)$ (9.22)	$r_{n+1}^{b}(N)$ (9.55)	(9.68)
$R_{n+1}^{-1}(N)$ (9.58)	$z_{n+1}^{T}(N)$ (9.23)	$\overline{r}_{n+1}(N)$ (9.56)	(9.69)
$R_{n+1}^{-1}(N)$ (9.59)	$z_{n+1}^{T}(N)$ (9.24)	$\overline{r}_{n+1}^{b}(N)$ (9.57)	(9.70)
$R_{n+1}^{-1}(N)$ (9.59)	$\varphi_{n+1}^{T}(N)$ (9.22)	$z_{n+1}(N)$ (9.24)	(9.71)
$R_{n+1}^{-1}(N)$ (9.58)	$\varphi_{n+1}^{T}(N)$ (9.21)	$z_{n+1}(N)$ (9.23)	(9.72)

Tabelul 9.2. Operaţii necesare pentru deducerea ecuaţiilor de recurenţă în ordin.

Aplicând matricei din membrul stâng al relaţiei (9.75) lema inversării matricelor partiţionate (Lema 3.2.2.), rezultă

$$\begin{bmatrix} R_n(N) & z_n(N) \\ \varphi_n^{T}(N) & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -R_n^{-1}(N)z_n(N) \\ 1 \end{bmatrix} \begin{bmatrix} -\varphi_n^{T}(N)R_n^{-1}(N) & 1 \end{bmatrix} \cdot$$

$$\cdot \left(1 - \varphi_n^{T}(N)R_n^{-1}(N)z_n(N)\right)^{-1} =$$

$$= \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} K_n^{*}(N) \\ 1 \end{bmatrix} \begin{bmatrix} \overline{K}_n^{*T}(N) & 1 \end{bmatrix} \frac{1}{\gamma_n^{0}(N)}. \tag{9.76}$$

Membrul drept al ecuaţiei (9.75) poate fi inversat direct:

$$\left\{ \begin{bmatrix} I & z_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_n(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \varphi_n^{T}(N) & 1 \end{bmatrix} \right\}^{-1} =$$

$$= \begin{bmatrix} I & 0 \\ \varphi_n^{T}(N) & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & z_n(N) \\ 0 & 1 \end{bmatrix}^{-1} =$$

$$= \begin{bmatrix} I & 0 \\ -\varphi_n^{T}(N) & 1 \end{bmatrix} \begin{bmatrix} R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix}. \tag{9.77}$$

Egalând expresiile (9.76) şi (9.77), rezultă ecuaţia de bază cu ajutorul căreia se vor deduce toate relaţiile recursive în timp

$$\begin{bmatrix} I & 0 \\ -\varphi_n^{T}(N) & 1 \end{bmatrix} \begin{bmatrix} R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \frac{1}{\gamma_n^{0}(N)} \begin{bmatrix} -R_n^{-1}(N)z_n(N) \\ 1 \end{bmatrix} \begin{bmatrix} -\varphi_n^{T}(N)R_n^{-1}(N) & 1 \end{bmatrix}. \tag{9.78}$$

Scanned with OKEN Scanner

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \frac{\alpha_n(N)}{\sigma_n^b(N)} \tag{9.61}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N-1) \end{bmatrix} - \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} \tag{9.62}$$

$$\overline{\theta}_{n+1}(N) = \begin{bmatrix} \overline{\theta}_n(N) \\ 0 \end{bmatrix} - \begin{bmatrix} \overline{\theta}_n^b(N) \\ 1 \end{bmatrix} \frac{\overline{\alpha}_n(N)}{\sigma_n^b(N)} \tag{9.63}$$

$$\overline{\theta}_{n+1}^b(N) = \begin{bmatrix} 0 \\ \overline{\theta}_n^b(N-1) \end{bmatrix} - \begin{bmatrix} 1 \\ \overline{\theta}_n(N-1) \end{bmatrix} \frac{\alpha_n(N-1)}{\sigma_n(N-1)} \tag{9.64}$$

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \frac{\overline{\varepsilon}_n^b(N)}{\sigma_n^b(N)} \tag{9.65}$$

$$K_{n+1}^*(N) = \begin{bmatrix} 0 \\ K_n^*(N-1) \end{bmatrix} - \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \frac{\overline{\varepsilon}_n(N-1)}{\sigma_n(N-1)} \tag{9.66}$$

$$\varepsilon_{n+1}(N) = \varepsilon_n(N) - \frac{\alpha_n(N)}{\sigma_n^b(N)} \varepsilon_n^b(N) \tag{9.67}$$

$$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) - \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} \varepsilon_n(N-1) \tag{9.68}$$

$$\overline{\varepsilon}_{n+1}(N) = \overline{\varepsilon}_n(N) - \frac{\overline{\alpha}_n(N)}{\sigma_n^b(N)} \overline{\varepsilon}_n^b(N) \tag{9.69}$$

$$\overline{\varepsilon}_{n+1}^b(N) = \overline{\varepsilon}_n^b(N-1) - \frac{\alpha_n(N-1)}{\sigma_n(N-1)} \overline{\varepsilon}_n(N-1) \tag{9.70}$$

$$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \varepsilon_n(N-1)\overline{\varepsilon}_n(N-1)/\sigma_n(N-1) \tag{9.71}$$

$$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \varepsilon_n^b(N)\overline{\varepsilon}_n^b(N)/\sigma_n^b(N) \tag{9.72}$$

$$\sigma_{n+1}(N) = \sigma_n(N) + \frac{\overline{\alpha}_n(N)\alpha_n(N)}{\sigma_n^b(N)} \tag{9.73}$$

$$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) + \frac{\overline{\alpha}_n(N-1)\alpha_n(N-1)}{\sigma_n(N-1)} \tag{9.74}$$

Tabelul 9.3. Relaţii de bază pentru recurenţa în ordin.

Scanned with OKEN Scanner

Să observăm că prima matrice din membrul stâng al ecuaţiei de mai sus realizează recursia în timp pentru vectorul $\bar{r}_n(N)$

$$\left[\bar{r}_n^{\mathrm{T}}(N)\ \ z(N)\right]\begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} = \left[\bar{r}_n^{\mathrm{T}}(N-1)\ \ z(N)\right] \tag{9.79}$$

şi pentru vectorul $\bar{r}_n^b(N)$

$$\left[\bar{r}_n^{b\mathrm{T}}(N)\ \ z(N-n-1)\right]\begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} = \left[\bar{r}_n^{b\mathrm{T}}(N-1)\ \ z(N-n-1)\right], \tag{9.80}$$

iar ultima matrice din membrul stâng al relaţiei (9.78) realizează recurenţa în timp pentru vectorul $r_n(N)$

$$\begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} r_n(N) \\ y(N) \end{bmatrix} = \begin{bmatrix} r_n(N-1) \\ y(N) \end{bmatrix} \tag{9.81}$$

şi pentru vectorul $r_n^b(N)$

$$\begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} r_n^b(N) \\ y(N-n-1) \end{bmatrix} = \begin{bmatrix} r_n^b(N-1) \\ y(N-n-1) \end{bmatrix}. \tag{9.82}$$

În continuare este ilustrat modul de deducere a relaţiilor de recurenţă în timp pentru un predictor, o eroare de predicţie şi o "dispersie", iar în Tabelul 9.4. sunt sintetizate înlocuirile necesare pentru obţinerea tuturor recursiilor în timp. În Tabelul 9.5. sunt prezentate recursiile în timp.

Înmulţind la dreapta membrul stâng al relaţiei (9.78) cu vectorul $\left[r_n^{\mathrm{T}}(N)\ \ y(N)\right]^{\mathrm{T}}$, rezultă

$$\begin{aligned} &\begin{bmatrix} R_n^{-1}(N-1) & 0 \\ -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1) & 1 \end{bmatrix}\begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} r_n(N) \\ y(N) \end{bmatrix} = \\ &= \begin{bmatrix} R_n^{-1}(N-1) & 0 \\ -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1) & 1 \end{bmatrix}\begin{bmatrix} r_n(N-1) \\ y(N) \end{bmatrix} = \\ &= \begin{bmatrix} R_n^{-1}(N-1)r_n(N-1) \\ y(N) - \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1)r_n(N-1) \end{bmatrix} = \begin{bmatrix} -\theta_n(N-1) \\ e_n(N) \end{bmatrix}, \end{aligned} \tag{9.83}$$

iar înmulţind cu acelaşi vector la dreapta şi membrul drept al relaţiei (9.78) rezultă

$$\begin{aligned} &\left\{\begin{bmatrix} R_n^{-1}(N) \\ 0 \end{bmatrix}\begin{bmatrix} I & 0 \end{bmatrix} + \frac{1}{\gamma_n^0(N)}\begin{bmatrix} K_n^*(N) \\ 1 \end{bmatrix}\begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix}\right\}\begin{bmatrix} r_n(N) \\ y(N) \end{bmatrix} = \\ &= \begin{bmatrix} R_n^{-1}(N)r_n(N) \\ 0 \end{bmatrix} + \frac{y(N) - \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N)r_n(N)}{\gamma_n^0(N)}\begin{bmatrix} K_n^*(N) \\ 1 \end{bmatrix} = \\ &= \begin{bmatrix} -\theta_n(N) \\ 0 \end{bmatrix} + \frac{\varepsilon_n(N)}{\gamma_n^0(N)}\begin{bmatrix} K_n^*(N) \\ 1 \end{bmatrix}. \end{aligned} \tag{9.84}$$

Egalând expresiile obţinute în (9.83) şi (9.84) rezultă

$$\begin{bmatrix} -\theta_n(N-1) \\ e_n(N) \end{bmatrix} = \begin{bmatrix} -\theta_n(N) + K_n^*(N)\varepsilon_n(N)/\gamma_n^0(N) \\ \varepsilon_n(N)/\gamma_n^0(N) \end{bmatrix}, \tag{9.85}$$

Scanned with OKEN Scanner

Ecuaţie de bază	Înmulţire la stânga	Înmulţire la dreapta	Ecuaţii rezultate
(9.78)		$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]^{\mathrm{T}}$	(9.89), (9.97)
(9.78)		$\left[r_n^{b\mathrm{T}}(N) \;\; y(N-n-1) \right]^{\mathrm{T}}$	(9.90), (9.98)
(9.78)	$\left[\overline{r}_n^{\mathrm{T}}(N) \;\; z(N) \right]$		(9.91), (9.99)
(9.78)	$\left[\overline{r}_n^{b\mathrm{T}}(N) \;\; z(N-n-1) \right]$		(9.92), (9.100)
(9.78)	$\left[\overline{r}_n^{\mathrm{T}}(N) \;\; z(N) \right]$	$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]^{\mathrm{T}}$	(9.93)
(9.78)	$\left[\overline{r}_n^{b\mathrm{T}}(N) \;\; z(N-n-1) \right]$	$\left[r_n^{b\mathrm{T}}(N) \;\; y(N-n-1) \right]^{\mathrm{T}}$	(9.94)
(9.78)	$\left[\overline{r}_n^{b\mathrm{T}}(N) \;\; z(N-n-1) \right]$	$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]^{\mathrm{T}}$	(9.95)
(9.78)	$\left[\overline{r}_n^{\mathrm{T}}(N) \;\; z(N) \right]$	$\left[r_n^{b\mathrm{T}}(N) \;\; y(N-n-1) \right]^{\mathrm{T}}$	(9.96)

Tabelul 9.4. Deducerea ecuaţiilor recursive în timp.

de unde se obţin imediat relaţiile (9.89) şi (9.97) din Tabelul 9.5.

Recursia în timp a "dispersiei" $\sigma_n(N)$ se obţine înmulţind relaţia (9.78) la dreapta cu $\left[r_n^{\mathrm{T}}(N) \;\; z(N) \right]^{\mathrm{T}}$ şi la stânga cu $\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]$. Se obţine astfel o ecuaţie având ca membru stâng

$$\left[\overline{r}_n^{\mathrm{T}}(N) \;\; z(N) \right] \begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -z_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y(N) \end{bmatrix} =$$
$$= \overline{r}_n^{\mathrm{T}}(N-1) R_n^{-1}(N-1) r_n(N-1) + z(N) y(N), \tag{9.86}$$

iar ca membru drept

$$\left[\overline{r}_n^{\mathrm{T}}(N) \;\; z(N) \right] \left\{ \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \left[I \;\; 0 \right] + \frac{1}{\gamma_n^0(N)} \begin{bmatrix} -R_n^{-1}(N) z_n(N) \\ 1 \end{bmatrix} \cdot \right.$$
$$\left. \cdot \left[-\varphi_n^{\mathrm{T}}(N) R_n^{-1}(N) \;\; 1 \right] \right\} \begin{bmatrix} r_n(N) \\ y(N) \end{bmatrix} = \overline{r}_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n(N) + \overline{\varepsilon}_n(N) \varepsilon_n(N) / \gamma_n^0(N). \tag{9.87}$$

Egalând (9.86) cu (9.87) şi scăzând din ambii membri $\sum_{t=0}^{N} z(t) y(t)$ rezultă

$$\overline{r}_n^{\mathrm{T}}(N-1) R_n^{-1}(N-1) r_n(N-1) - \sum_{t=0}^{N-1} z(t) y(t) =$$
$$= \overline{r}_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n(N) - \sum_{t=0}^{N} z(t) y(t) + \overline{\varepsilon}_n(N) \varepsilon_n(N) / \gamma_n^0(N), \tag{9.88}$$

din care rezultă imediat relaţia (9.93) din Tabelul 9.5.

Algoritmul latice pentru erorile de predicţie

Vom selecta un set închis de relaţii dintre ecuaţiile recursive în ordin (Tabelul 9.3.) şi dintre ecuaţiile recursive în timp (Tabelul 9.5.) prin care să se propage în timp întreaga informaţie

Scanned with OKEN Scanner

$$\theta_n(N) = \theta_n(N-1) + \frac{\varepsilon_n(N)}{\gamma_n^0(N)} K_n^*(N) \tag{9.89}$$

$$\theta_n^b(N) = \theta_n^b(N-1) + \frac{\varepsilon_n^b(N)}{\gamma_n^0(N)} K_n^*(N) \tag{9.90}$$

$$\overline{\theta}_n(N) = \overline{\theta}_n(N-1) + \frac{\overline{\varepsilon}_n(N)}{\gamma_n^0(N)} \overline{K}_n^*(N) \tag{9.91}$$

$$\overline{\theta}_n^b(N) = \overline{\theta}_n^b(N-1) + \frac{\overline{\varepsilon}_n^b(N)}{\gamma_n^0(N)} \overline{K}_n^*(N) \tag{9.92}$$

$$\sigma_n(N) = \sigma_n(N-1) + \frac{\overline{\varepsilon}_n(N)\varepsilon_n(N)}{\gamma_n^0(N)} \tag{9.93}$$

$$\sigma_n^b(N) = \sigma_n^b(N-1) + \frac{\overline{\varepsilon}_n^b(N)\varepsilon_n^b(N)}{\gamma_n^0(N)} \tag{9.94}$$

$$\alpha_n(N) = \alpha_n(N-1) + \frac{\overline{\varepsilon}_n^b(N)\varepsilon_n(N)}{\gamma_n^0(N)} \tag{9.95}$$

$$\overline{\alpha}_n(N) = \overline{\alpha}_n(N-1) + \frac{\overline{\varepsilon}_n(N)\varepsilon_n^b(N)}{\gamma_n^0(N)} \tag{9.96}$$

$$\varepsilon_n(N) = e_n(N)\gamma_n^0(N) \tag{9.97}$$

$$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N) \tag{9.98}$$

$$\overline{\varepsilon}_n(N) = \overline{e}_n(N)\gamma_n^0(N) \tag{9.99}$$

$$\overline{\varepsilon}_n^b(N) = \overline{e}_n^b(N)\gamma_n^0(N) \tag{9.100}$$

Tabelul 9.5. Relaţii de bază pentru recurenţa în timp.

Scanned with OKEN Scanner

necesară pentru determinarea la un moment de timp la alegere a parametrilor secţiunii AR a modelului ARMA.

În Tabelul 9.6. este prezentată una dintre variantele de alegeri posibile ale unui set complet de ecuaţii recursive în care se remarcă trei secţiuni latice: $\{\sigma_n, \sigma_n^b\}$, $\{\varepsilon_n, \varepsilon_n^b\}$ şi $\{\overline{\varepsilon}_n, \overline{\varepsilon}_n^b\}$, fiind necesare în plus două recursii în timp pentru intercovarianţele α_n şi $\overline{\alpha}_n$ şi o recursie în ordin pentru raportul aposteriori/ apriori, γ_n^0. Iniţializarea recursiilor în ordin, pentru ordinul fictiv $n = 0$, rezultă imediat din definiţiile din Tabelul 9.1.

Se constată că structura algoritmului este foarte asemănătoare cu cea a Algoritmului 7.3.1.; se poate demonstra că tratarea cazului matricei $R_n(N)$ singulare (care în algoritmul latice se reflectă prin $\sigma_n = 0$ sau $\sigma_n^b = 0$) este perfect similară cazului singular tratat în Capitolul 8 (în relaţiile în care apar împărţiri la σ_n sau σ_n^b se va atribui valoarea zero rezultatului dacă împărţitorul este zero).

Numărul de operaţii implicate în algoritmul 9.1 este mai mare decât cel necesar în secţiunea latice din Algoritmul 7.4, dar ordinul de mărime rămâne acelaşi: $\mathcal{O}(n_{max})$.

În cazul în care se ia $z(t) = y(t)$, deci când modelul ARMA se reduce la un model AR, toate mărimile suprabarate din Algoritmul 9.1 devin egale cu cele nebarate, algoritmul devenind identic cu secţiunea latice a Algoritmului 7.4.

Sunt posibile şi alte variante ale Algoritmului 9.1, dar numărul necesar de operaţii nu va diferi semnificativ. De exemplu, ecuaţiile (5) şi (6) din Algoritmul 9.1 pot fi înlocuite cu ecuaţiile (9.93) şi (9.94) (totuşi prima variantă este mai convenabilă pentru că iniţializarea ecuaţiilor recursive în timp este mai dificilă decât iniţializarea ecuaţiilor recursive în ordin).

9.2. Trecerea de la parametrii latice la parametrii AR

Algoritmul latice al erorilor propagă în timp informaţia sub forma unor secţiuni latice, ai căror coeficienţi de reflexie sunt adaptaţi la fiecare pas pentru a fi satisfăcute exact ecuaţiile (9.17) pentru toate ordinele $n = 1, \ldots, n_{max}$. Laticea erorilor de predicţie corespunzătoare predictorilor $\theta_n(N)$ şi $\theta_n^b(N)$ este dată de ecuaţiile (8) şi (9) din Tabelul 9.6. Coeficienţii de reflexie ai acestei secţiuni sunt:

$$K_n^f(N) = \frac{\alpha_n(N)}{\sigma_n^b(N)}, \tag{9.101}$$

$$K_n^b(N) = \frac{\overline{\alpha}_n(N)}{\sigma_n(N)}. \tag{9.102}$$

Trecerea de la această parametrizare latice la parametrizarea directă AR se poate realiza printr-o recursie de tip Levinson, care rezultă din ecuaţiile recursive în ordin (9.61) şi (9.62) din Tabelul 9.3. Ecuaţia (9.62) nu este totuşi o recursie pură în ordin, datorită schimbării argumentului N în $N-1$. De aceea, pentru a realiza trecerea inversă, de la argumentul $N-1$ la N, vor fi utilizate ecuaţiile recurente în timp (9.89) şi (9.90). În acest mod, relaţia (9.62) se poate transforma:

Scanned with OKEN Scanner

Algoritmul 9.1.1.

ALGORITM LATICE VI PENTRU ERORILE DE PREDICŢIE		*Număr operaţii/ ecuaţie* ⋆ / +		*Număr operaţii/ pas alg.*
1. *Iniţializare la momentul* $(N = 0)$				
Pentru $n = 0, \ldots, n_{max} - 1$				
$\overline{\alpha}_n(0) = 0; \quad \alpha_n(0) = 0$				
$\varepsilon_n(0) = \varepsilon_n^b(0) = \overline{\varepsilon}_n(0) = \overline{\varepsilon}_n^b(0) = 0$				
$\sigma_0(0) = \sigma_0^b(0) = 0$				
2. *Recursii în timp*				$\mathcal{O}(n_{max}^2)$
Pentru $N = 1, 2, \ldots, N_{max}$				
2.1 *Iniţializare recursii în ordin*				
$\varepsilon_0(N) = y(N); \quad \varepsilon_0^b(N) = y(N-1)$				
$\overline{\varepsilon}_0(N) = z(N); \quad \overline{\varepsilon}_0^b(N) = z(N-1)$				
$\gamma_0^0(N) = 1$				
$\sigma_0(N) = \sigma_0(N-1) + y(N)z(N)$	(1)	1	0 1	
$\sigma_0^b(N) = \sigma_0(N-1)$	(2)			
2.2 *Recursii în ordin:*				$\mathcal{O}(n_{max})$
Pentru $n = 0, \ldots, n_{max} - 1$				
$\alpha_n(N) = \alpha_n(N-1) + \overline{\varepsilon}_n^b(N)\varepsilon_n(N)/\gamma_n^0(N)$	(3)	1	1 1	
$\overline{\alpha}_n(N) = \overline{\alpha}_n(N-1) + \overline{\varepsilon}_n(N)\varepsilon_n^b(N)/\gamma_n^0(N)$	(4)	1	1 1	
$\sigma_{n+1}(N) = \sigma_n(N) - \alpha_n(N)\overline{\alpha}_n(N)/\sigma_n^b(N)$	(5)	1	1 1	
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \alpha_n(N-1)\overline{\alpha}_n(N-1)/\sigma_n(N-1)$	(6)	1	1 1	
$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \overline{\varepsilon}_n^b(N)\varepsilon_n(N)/\sigma_n^b(N)$	(7)	1	1 1	
$\varepsilon_{n+1}(N) = \varepsilon_n(N) - \alpha_n(N)\varepsilon_n^b(N)/\sigma_n^b(N)$	(8)	1	1 1	
$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) - \overline{\alpha}_n(N-1)\varepsilon_n(N-1)/\sigma_n(N-1)$	(9)	1	1 1	
$\overline{\varepsilon}_{n+1}(N) = \overline{\varepsilon}_n(N) - \overline{\alpha}_n(N)\overline{\varepsilon}_n^b(N)/\sigma_n^b(N)$	(10)	1	1 1	
$\overline{\varepsilon}_{n+1}^b(N) = \overline{\varepsilon}_n^b(N-1) - \alpha_n(N-1)\overline{\varepsilon}_n(N-1)/\sigma_n(N-1)$	(11)	1	1 1	
$(\star)$: *Dacă împărţitorul* $\delta = 0$ *se face atribuirea* $\frac{1}{\delta} \leftarrow 0$				
Total: $9n_{max}$ *înmulţiri;* $9n_{max}$ *împărţiri;*$9n_{max}$ *adunări*				

Tabelul 9.6. Algoritmul "latice VI a erorilor de predicţie".

Scanned with OKEN Scanner

SECŢIUNEA OPŢIONALĂ PENTRU CALCULUL PREDICTORILOR

2. Recursii în timp

2.3 Dacă la pasul N se doreşte evaluarea predictorilor

Atunci:

2.3.1 Iniţializare recursii în ordin:

$$\theta_1(N) = -\frac{\alpha_0(N)}{\sigma_0^b(N)}$$

$$\theta_1^b(N) = -\frac{\overline{\alpha}_0(N-1)}{\sigma_0(N-1)}$$

$$K_1^*(N) = -\frac{\overline{\varepsilon}_0^b(N)}{\sigma_0^b(N)}$$

2.3.2 Recursii în ordin:

Pentru $n = 1, \ldots, n_{max} - 1$

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} - \frac{\alpha_n(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} - \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} - \frac{\varepsilon_n^b(N) - \varepsilon_n(N)\overline{\alpha}_n(N-1)/\sigma_n(N-1)}{\gamma_n^0(N)} \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix}$$

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{\overline{\varepsilon}_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix}$$

Total: $\mathcal{O}(n_{max}^2)$ flops

Tabelul 9.7. Extinderea Algoritmului 9.1.1 cu o secţiune pentru trecerea de la erorile de predicţie la predictori.

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N-1) \end{bmatrix} - \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} =$$

$$= \begin{bmatrix} 0 \\ \theta_n^b(N) - K_n^*(N)\varepsilon_n^b(N)/\gamma_n^0(N) \end{bmatrix} - \begin{bmatrix} 1 \\ \theta_n(N) - K_n^*(N)\varepsilon_n(N)/\gamma_n^0(N) \end{bmatrix} \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} =$$

$$= \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} - \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} \frac{\overline{\alpha}_n(N-1)}{\sigma_n(N-1)} - \begin{bmatrix} 1 \\ K_n^*(N) \end{bmatrix} \frac{\varepsilon_n^b(N) - \varepsilon_n(N)\overline{\alpha}_n(N-1)/\sigma_n(N-1)}{\gamma_n^0(N)}.$$

Un set complet de recursii se obţine adăugând relaţiilor (9.61),(9.103), relaţia (9.65).

În Tabelul 9.7. este prezentată secţiunea recursivă în ordin pentru calculul parametrilor AR ai modelului ARMA, care va fi executată opţional, numai atunci când se doreşte evaluarea parametrilor AR. Numărul de operaţii necesar este de ordinul $\mathcal{O}(n_{max}^2)$.

Scanned with OKEN Scanner

Scanned with OKEN Scanner

Capitolul 10

Algoritmul Latice pentru Modele Multivariabile de Tip Sumă de Convoluţie

10.1. Algoritmul latice pentru erorile de predicţie

În acest capitol vor fi prezentate metode rapide de estimare a parametrilor modelelor multivariabile (în parametrizarea latice şi în cea directă) de tip sumă de convoluţie. Această clasă de modele este foarte largă, cuprinzând şi modele multivariablie AR şi modele multivariabile de tip ecuaţie cu diferenţe. Ca metodă de estimare va fi utilizată metoda CMMP cu criteriu determinist, în care va fi introdus şi un factor de ponderare pentru a crea posibilitatea urmăririi parametrilor supuşi variaţiei lente în timp. Având în vedere că există o multitudine de variante de algoritmi, mai întâi va fi abordată deducerea sistematică a tuturor relaţiilor recurente în ordin şi în timp pentru parametrii modelelor, predicţiile modelelor şi pentru alte mărimi auxiliare necesare obţinerii de seturi închise de relaţii recursive (de notat că aceste seturi de ecuaţii vor fi utilizate şi în Capitolele 11 şi 12, în care vor fi abordate alte clase de algoritmi, decât cei latice, prezentaţi în acest capitol). Se vor evidenţia apoi structurile recursive în ordin (de tip latice) prin care se obţine parametrizarea latice a modelelor de tip sumă de convoluţie. Trecerea de la această parametrizare la parametrizarea directă va fi abordată în Secţiunea 10.2.

Fie modelul multivariabil de tip sumă de convoluţie

$$y(t) = \sum_{i=1}^{n} \theta_{n,i}^{\mathrm{T}} u(t-i) + v_n(t), \tag{10.1}$$

unde:

$u(t)$ este vectorul intrărilor, $q-$ dimensional;

$y(t)$ este vectorul ieşirilor, $p-$ dimensional;

$v_n(t)$ este vectorul perturbaţiilor, $p-$ dimensional;

$\theta_{n,i}$ sunt matricele parametrilor modelului, $q \times p-$ dimensionale.

Vom nota vectorul regresorilor, de dimensiune $n \cdot q$,

$$\varphi_n(t) = \begin{bmatrix} u(t-1) \\ \cdot \\ \cdot \\ \cdot \\ u(t-n) \end{bmatrix} \tag{10.2}$$

şi matricea parametrilor, de dimensiuni $p \times nq$,

$$\theta_n = \begin{bmatrix} \theta_{n,1} \\ \cdot \\ \cdot \\ \theta_{n,n} \end{bmatrix} \tag{10.3}$$

rezultând o altă formă a ecuaţiei modelului (10.1),

$$y(t) = \theta_n^{\mathrm{T}} \varphi_n(t) + v_n(t). \tag{10.4}$$

Ecuaţiile (10.1) sau (10.4) acoperă o clasă foarte largă de modele. Astfel, dacă

$$u(t) = y(t) \tag{10.5}$$

va rezulta modelul AR multivariabil

$$y(t) - \sum_{i=1}^{n} \theta_{n,i} y(t-i) = v_n(t). \tag{10.6}$$

Modelul multivariabil de tip ecuaţie cu diferenţe

$$y^{[1]}(t) + \sum_{i=1}^{n} A_{n,i}^{\mathrm{T}} y^{[1]}(t-i) = \sum_{i=1}^{n} B_{n,i}^{\mathrm{T}} u^{[1]}(t-i) + v_n(t) \tag{10.7}$$

poate fi adus la forma (10.1) adoptând notaţiile:

$$y(t) = y^{[1]}(t), \tag{10.8}$$

$$u(t) = \begin{bmatrix} -y^{[1]}(t) \\ u^{[1]}(t) \end{bmatrix}, \tag{10.9}$$

$$\theta_{n,i}^{\mathrm{T}} = \begin{bmatrix} A_{n,i}^{\mathrm{T}} B_{n,i}^{\mathrm{T}} \end{bmatrix}. \tag{10.10}$$

În consecinţă, algoritmii rapizi deduşi în acest capitol pentru modelul (10.1) vor furniza imediat (după eliminarea unor ecuaţii redundante în cazul particular (10.5)) algoritmi rapizi şi pentru modelele (10.6) şi (10.7).

În Capitolul 5 a fost rezolvată problema estimării unui model de tip (10.1) prin metoda CMMP, în care criteriul minimizat a fost o funcţie scalară având ca argument matricea de covarianţă a zgomotului $v_n(t)$. Implementarea optimală a soluţiei ar cere cunoaşterea unor funcţii teoretice de covarianţă; atunci când datele disponibile sunt valorile seriei intrărilor $\{u(t)\}_{t=0}^{N}$ şi ale seriei ieşirilor $\{y(t)\}_{t=0}^{N}$, se poate obţine numai o implementare suboptimală a soluţiei prin înlocuirea funcţiilor teoretice de covarianţă cu covarianţele de eşantion.

Scanned with OKEN Scanner

În acest capitol vor fi obţinute estimaţii ale parametrilor modelului (10.1) prin minimizarea exactă a unui criteriu de tipul CMMP determinist, de fapt prin minimizarea matricei de covarianţă de eşantion a zgomotului, dar cu deponderarea exponenţială a datelor "vechi". Se va adopta ipoteza "fereastră anterioară",

$$y(t) = 0, \quad u(t) = 0 \qquad \text{pentru } t < 0, \tag{10.11}$$

ipoteză care va simplifica structurile partiţionate (a căror exploatare constituie cheia obţinerii unor algoritmi rapizi).

Criteriul matriceal a cărui minimizare este abordată în continuare este definit astfel:

$$\begin{aligned} Q_{n,N}(\theta_n) &= \sum_{t=0}^{N} \lambda^{N-t} v_n(t) v_n^{\mathrm{T}}(t) = \\ &= \sum_{t=0}^{N} \lambda^{N-t} (y(t) - \theta_n^{\mathrm{T}} \varphi_n(t))(y(t) - \theta_n^{\mathrm{T}} \varphi_n(t))^{\mathrm{T}}, \end{aligned} \tag{10.12}$$

în care $\lambda \in \{0, 1\}$ este factorul de deponderare exponenţială.

Introducând notaţiile:

$$R_n(N) = \sum_{t=0}^{N} \lambda^{N-t} \varphi_n(t) \varphi_n^{\mathrm{T}}(t) \quad \text{(matrice } nq \times nq), \tag{10.13}$$

$$r_n(N) = \sum_{t=0}^{N} \lambda^{N-t} \varphi_n(t) y^{\mathrm{T}}(t) \quad \text{(matrice } nq \times p), \tag{10.14}$$

criteriul (10.12) poate fi scris

$$\begin{aligned} Q_{n,N}(\theta_n) &= \sum_{t=0}^{N} \lambda^{N-t} (y(t) - \theta_n^{\mathrm{T}} \varphi_n(t))(y(t) - \theta_n^{\mathrm{T}} \varphi_n(t))^{\mathrm{T}} = \\ &= \sum_{t=0}^{N} \lambda^{N-t} y(t) y^{\mathrm{T}}(t) - \theta_n^{\mathrm{T}} r_n(N) - r_n^{\mathrm{T}}(N) \theta_n + \theta_n^{\mathrm{T}} R_n(N) \theta_n = \\ &= (\theta_n - R_n^{-1}(N) r_n(N))^{\mathrm{T}} R_n(N) (\theta_n - R_n^{-1}(N) r_n(N)) + \sum_{t=0}^{N} \lambda^{N-t} y(t) y^{\mathrm{T}}(t) - \\ &\quad - r_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n(N). \end{aligned} \tag{10.15}$$

Întrucât matricea $R_n(N)$ este pozitiv semidefinită (vezi demonstraţia din Capitolul 7), rezultă că primul termen din ultimul membru al relaţiei (10.15) este pozitiv sau zero. Notând

$$\theta_n(N) = R_n^{-1}(N) r_n(N) \tag{10.16}$$

şi

$$\sigma_n(N) = Q_{n,N}(\theta_n(N)) = \sum_{t=0}^{N} \lambda^{N-t} y(t) y^{\mathrm{T}}(t) - r_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n(N), \tag{10.17}$$

va rezulta

$$Q_{n,N}(\theta_n) - Q_{n,N}(\theta_n(N)) = Q_{n,N}(\theta_n) - \sigma_n(N) \geq 0,$$

Scanned with OKEN Scanner

deci $\theta_n(N)$ va minimiza criteriul matriceal (10.15).

Numărul de operaţii implicat în rezolvarea sistemului (10.16) este $\mathcal{O}(n^3q^3)$, iar dacă se cer toate matricele de parametri $\{\theta_n(N)\}_{n=1}^{n_{max}}$ (aşa cum este în aplicaţiile în care se urmăreşte şi estimarea ordinului modelului), atunci sunt necesare $\mathcal{O}(n_{max}^4q^3)$ operaţii. Algoritmii din acest capitol estimează întreg setul de modele $\{\theta_n(N)\}_{n=1}^{n_{max}}$ în $\mathcal{O}(n_{max}^2q^3)$ operaţii/pas de eşantionare sau pot propaga parametrii latice corespunzători întregului set de modele în $\mathcal{O}(n_{max}q^3)$ operaţii/pas de eşantionare.

Ca şi în capitolele precedente, recurenţa în ordin pentru mărimile implicate în (10.16) va fi posibilă datorită structurii tip "registru de deplasare" a vectorului $\varphi_n(t)$ definit în (10.2). Astfel, partiţiile vectorului $\varphi_{n+1}(t)$

$$\varphi_{n+1}(t) = \begin{bmatrix} \varphi_n(t) \\ u(t-n-1) \end{bmatrix} = \tag{10.18}$$

$$= \begin{bmatrix} u(t-1) \\ \varphi_n(t-1) \end{bmatrix} \tag{10.19}$$

vor determina următoarele structuri partiţionate ale mărimilor necesare pentru calculul soluţiei (10.16)

$$r_{n+1}(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_{n+1}(t)y^{\mathrm{T}}(t) = \begin{bmatrix} r_n(N) \\ \sum_{t=0}^{N} \lambda^{N-t}u(t-n-1)y^{\mathrm{T}}(t) \end{bmatrix} =$$

$$= \begin{bmatrix} r_n(N) \\ \rho_{n+1}^{yu}(N) \end{bmatrix}, \tag{10.20}$$

$$R_{n+1}(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_{n+1}(t)\varphi_{n+1}^{\mathrm{T}}(t) = \begin{bmatrix} R_n(N) & r_n^b(N) \\ r_n^{b\mathrm{T}}(N) & \rho_0^u(N-n-1) \end{bmatrix} = \tag{10.21}$$

$$= \begin{bmatrix} \rho_0^u(N-1) & r_n^{f\mathrm{T}}(N-1) \\ r_n^f(N-1) & R_n(N-1) \end{bmatrix}. \tag{10.22}$$

În expresiile de mai sus au fost utilizate notaţiile:

$$r_n^f(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)u^{\mathrm{T}}(t), \tag{10.23}$$

$$r_n^b(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)u^{\mathrm{T}}(t-n-1), \tag{10.24}$$

$$\rho_m^{yu}(N) = \sum_{t=0}^{N} \lambda^{N-t}u(t-m)y^{\mathrm{T}}(t); \quad \rho_m^u(N) = \rho_m^{uu}(N). \tag{10.25}$$

Aplicând matricelor (10.21) şi (10.22) lema inversării matricelor partiţionate se obţin următoarele expresii pentru $R_{n+1}^{-1}(N)$:

$$R_{n+1}^{-1}(N) = \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -R_n^{-1}(N)r_n^b(N) \\ I \end{bmatrix} \cdot$$

$$\cdot(\rho_0^u(N-n-1) - r_n^{b\mathrm{T}}(N)R_n^{-1}(N)r_n^b(N))^{-1}\begin{bmatrix} -r_n^{b\mathrm{T}}(N)R_n^{-1}(N) & I \end{bmatrix} =$$

Scanned with OKEN Scanner

$$= \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \begin{bmatrix} -\theta_n^{b\mathrm{T}}(N) & I \end{bmatrix}, \tag{10.26}$$

$$R_{n+1}^{-1}(N) = \begin{bmatrix} 0 & 0 \\ 0 & R_n^{-1}(N-1) \end{bmatrix} + \begin{bmatrix} I \\ -R_n^{-1}(N-1)r_n^f(N-1) \end{bmatrix} \cdot$$
$$\cdot(\rho_o^u(N-1) - r_N^{f\mathrm{T}}(N-1)R_n^{-1}(N-1)r_n^f(N-1))^{-1} \begin{bmatrix} I & -r_n^{f\mathrm{T}}(N-1)R_n^{-1}(N-1) \end{bmatrix}$$

$$R_{n+1}^{-1}(N) = \begin{bmatrix} 0 & 0 \\ 0 & R_n^{-1}(N-1) \end{bmatrix} + \begin{bmatrix} I \\ -\theta_n^f(N-1) \end{bmatrix} (\sigma_n^f(N-1))^{-1} \begin{bmatrix} I & -\theta_n^{f\mathrm{T}}(N-1) \end{bmatrix}. \tag{10.27}$$

În aceste expresii intervin mărimile auxiliare:

$$\theta_n^f(N) = R_n^{-1}(N)r_n^f(N), \tag{10.28}$$
$$\theta_n^b(N) = R_n^{-1}(N)r_n^b(N), \tag{10.29}$$
$$\sigma_n^f(N) = \rho_0^u(N) - r_n^{f\mathrm{T}}(N)R_n^{-1}(N)r_n^f(N), \tag{10.30}$$
$$\sigma_n^b(N) = \rho_0^u(N-n-1) - r_n^{b\mathrm{T}}(N)R_n^{-1}(N)r_n^b(N). \tag{10.31}$$

Comparând definiţiile mărimilor auxiliare (10.23),(10.24),(10.28)–(10.31) cu definiţiile (7.35), (7.36) şi (7.38)–(7.41), se observă că pentru cazul $q = 1$, θ_n^f şi θ_n^b din (10.28) şi (10.29) sunt predictorii înainte, respectiv înapoi, optimali în sensul criteriului CMMP ponderat, corespunzători semnalului $u(t)$, iar σ_n^f şi σ_n^b din (10.30) şi (10.31) sunt valorile minime ale dispersiilor de eşantion ale predicţiilor optimale înainte, respectiv înapoi, deci mărimile auxiliare introduse realizează modelarea AR a semnalului $u(t)$. Această interpretare rămâne valabilă şi în cazul multivariabil, predictorii (10.28) şi (10.29) fiind argumentele care minimizează criteriile matriceale (dispersiile de eşantion ale erorilor de predicţie):

$$Q_{n,N}^f(\theta_n^f) = \sum_{t=0}^{N} \lambda^{N-t}(u(t) - \theta_n^{f\mathrm{T}}\varphi_n(t))(u(t) - \theta_n^{f\mathrm{T}}\varphi_n(t))^{\mathrm{T}}, \tag{10.32}$$

$$Q_{n,N}^b(\theta_n^b) = \sum_{t=0}^{N} \lambda^{N-t}(u(t-n-1) - \theta_n^{b\mathrm{T}}\varphi_n(t))(u(t-n-1) - \theta_n^{b\mathrm{T}}\varphi_n(t))^{\mathrm{T}}, \tag{10.33}$$

iar valorile minime ale criteriilor (10.32) şi (10.33) sunt:

$$Q_{n,N}^f(\theta_n^f(N)) = \sigma_n^f(N), \tag{10.34}$$
$$Q_{n,N}^b(\theta_n^b(N)) = \sigma_n^b(N), \tag{10.35}$$

aşa cum rezultă imediat dezvoltând cele două criterii ca în relaţia (10.15) şi invocând semipozitiv-definirea matricei $R_n(N)$.

Vom defini în continuare erorile de predicţie apriori, corespunzătoare predicţiei utilizând observaţiile $\varphi_n(N)$ şi predictorii calculaţi la momentul $N-1$:

$$e_n(N) = y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N), \tag{10.36}$$
$$e_n^f(N) = u(N) - \theta_n^{f\mathrm{T}}(N-1)\varphi_n(N), \tag{10.37}$$
$$e_n^b(N) = u(N-n-1) - \theta_n^{b\mathrm{T}}(N-1)\varphi_n(N), \tag{10.38}$$

Scanned with OKEN Scanner

şi erorile de predicţie aposteriori, corespunzătoare predicţiei utilizând observaţiile $\varphi_n(t)$ şi predictorii calculaţi la momentul N:

$$\varepsilon_n(N) = y(N) - \theta_n^{\mathrm{T}}(N)\varphi_n(N), \tag{10.39}$$

$$\varepsilon_n^f(N) = u(N) - \theta_n^{f\mathrm{T}}(N)\varphi_n(N), \tag{10.40}$$

$$\varepsilon_n^b(N) = u(N-n-1) - \theta_n^{b\mathrm{T}}(N)\varphi_n(N). \tag{10.41}$$

Ca mărimi auxiliare în algoritmii rapizi vor mai fi utilizate:

- matricele de intercovarianţă de eşantion a erorilor de predicţie:

$$\begin{aligned}\alpha_n^y(N) &= \sum_{t=0}^{N}\lambda^{N-t}(y(t) - \theta_n^{\mathrm{T}}(N)\varphi_n(t))(u(t-n-1) - \theta_n^{b\mathrm{T}}(N)\varphi_n(t))^{\mathrm{T}} = \\ &= \sum_{t=0}^{N}\lambda^{N-t}y(t)u^{\mathrm{T}}(t-n-1) - r_n^{\mathrm{T}}(N)\theta_n^b(N) - \theta_n^{\mathrm{T}}(N)r_n^b(N) + \\ &+ \theta_n(N)R_n(N)\theta_n(N) = (\rho_{n+1}^{yu}(N))^{\mathrm{T}} - r_n^{\mathrm{T}}(N)R_n^{-1}(N)r_n^b(N),\end{aligned} \tag{10.42}$$

$$\begin{aligned}\alpha_n(N) &= \sum_{t=0}^{N}\lambda^{N-t}(u(t-n-1) - \theta_n^{b\mathrm{T}}(N)\varphi_n(t))(u(t) - \theta_n^{f\mathrm{T}}(N)\varphi_n(t))^{\mathrm{T}} = \\ &= \rho_{n+1}^u(N) - r_n^{b\mathrm{T}}(N)R_n^{-1}(N)r_n^f(N);\end{aligned} \tag{10.43}$$

- vectorul amplificărilor Kalman apriori

$$K_n(N) = \frac{1}{\lambda}R_n^{-1}(N-1)\varphi_n(N); \tag{10.44}$$

- vectorul amplificărilor Kalman aposteriori

$$K_n^*(N) = R_n^{-1}(N)\varphi_n(N); \tag{10.45}$$

- raportul aposteriori/ apriori

$$\gamma_n^0(N) = 1 - \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N)\varphi_n(N). \tag{10.46}$$

Relaţii recursive în ordin

Utilizând relaţiile (10.18),(10.19) şi relaţiile de definiţie (10.23),(10.24) rezultă

$$\begin{aligned}r_{n+1}^f(N) &= \sum_{t=0}^{N}\lambda^{N-t}\varphi_{n+1}(t)u^{\mathrm{T}}(t) = \\ &= \begin{bmatrix} r_n^f(N) \\ \sum_{t=0}^{N}\lambda^{N-t}u(t-n-1)u^{\mathrm{T}}(t)\end{bmatrix} = \begin{bmatrix} r_n^f(N) \\ \rho_{n+1}^u(N)\end{bmatrix},\end{aligned} \tag{10.47}$$

$$\begin{aligned}r_{n+1}^b(N) &= \sum_{t=0}^{N}\lambda^{N-t}\varphi_{n+1}(t)u^{\mathrm{T}}(t-n-2) = \\ &= \begin{bmatrix} \sum_{t=0}^{N}\lambda^{N-t}u(t-1)u^{\mathrm{T}}(t-n-2) \\ \sum_{t=0}^{N}\lambda^{N-t}\varphi_n(t-1)u^{\mathrm{T}}(t-n-2)\end{bmatrix} = \begin{bmatrix} \rho_{n+1}^{uT}(N-1) \\ r_n^b(N-1)\end{bmatrix}.\end{aligned} \tag{10.48}$$

Scanned with OKEN Scanner

Partiţiile (10.18)–(10.20),(10.47),(10.48) împreună cu ecuaţiile (10.26),(10.27) generează, prin simpla înlocuire în definiţiile mărimilor de ordin $n+1$ relaţiile de legătură cu mărimile de ordin n.

Vom nota în primul rând existenţa următoarelor identităţi:

$$\varepsilon_n^b(N) = \left[-\theta_n^{b\mathrm{T}}(N) \;\; I \right] \begin{bmatrix} \varphi_n(N) \\ u(N-n-1) \end{bmatrix} = \left[-\theta_n^{b\mathrm{T}}(N) \;\; I \right] \varphi_{n+1}(N), \tag{10.49}$$

$$e_n^b(N) = \left[-\theta_n^{b\mathrm{T}}(N-1) \;\; I \right] \begin{bmatrix} \varphi_n(N) \\ u(N-n-1) \end{bmatrix} = \left[-\theta_n^{b\mathrm{T}}(N-1) \;\; I \right] \varphi_{n+1}(N), \tag{10.50}$$

$$\varepsilon_n^f(N) = \left[I \;\; -\theta_n^{f\mathrm{T}}(N) \right] \begin{bmatrix} u(N) \\ \varphi_n(N) \end{bmatrix} = \left[I - \theta_n^{f\mathrm{T}}(N) \right] \varphi_{n+1}(N+1), \tag{10.51}$$

$$e_n^f(N) = \left[I \;\; -\theta_n^{f\mathrm{T}}(N-1) \right] \begin{bmatrix} u(N) \\ \varphi_n(N) \end{bmatrix} = \left[I \;\; -\theta_n^{f\mathrm{T}}(N-1) \right] \varphi_{n+1}(N+1), \tag{10.52}$$

$$\begin{aligned} \alpha_n^y(N) &= (\rho_{n+1}^{yu}(N))^{\mathrm{T}} - r_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n^b(N) = \\ &= \left[r_n^{\mathrm{T}}(N) \;\; (\rho_{n+1}^{yu}(N))^{\mathrm{T}} \right] \begin{bmatrix} -R_n^{-1}(N) r_n^b(N) \\ I \end{bmatrix} = r_{n+1}^{\mathrm{T}}(N) \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix}, \end{aligned} \tag{10.53}$$

$$\begin{aligned} \alpha_n(N) &= \rho_{n+1}^u(N) - r_n^{b\mathrm{T}}(N) R_n^{-1}(N) r_n^f(N) = \\ &= \left[-r_n^{b\mathrm{T}}(N) R_n^{-1}(N) \;\; I \right] \begin{bmatrix} r_n^f(N) \\ \rho_{n+1}^u(N) \end{bmatrix} = \left[-\theta_n^{b\mathrm{T}}(N) \;\; I \right] r_{n+1}^f(N) = \end{aligned} \tag{10.54}$$

$$= \left[\rho_{n+1}^u(N) \;\; r_n^{b\mathrm{T}}(N) \right] \begin{bmatrix} I \\ -R_n^{-1}(N) r_n^f(N) \end{bmatrix} = r_{n+1}^{b\mathrm{T}}(N+1) \begin{bmatrix} I \\ -\theta_n^f(N) \end{bmatrix}. \tag{10.55}$$

Utilizând identităţile de mai sus şi urmând modul de deducere prezentat în Tabelul 10.1., vor rezulta ecuaţiile recursive în ordin din Tabelul 10.2.

De exemplu, înmulţind la dreapta ambii membri ai ecuaţiei (10.26) cu $r_{n+1}(N)$ partiţionat ca în (10.20) şi utilizând definiţia predictorului $\theta_{n+1}(N)$ (10.16) şi identitatea (10.53) va rezulta

$$\begin{aligned} \theta_{n+1}(N) &= R_{n+1}^{-1}(N) r_{n+1}(N) = \\ &= \left\{ \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \left[-\theta_n^{b\mathrm{T}}(N) \;\; I \right] \right\} r_{n+1}(N) = \\ &= \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \alpha_n^{y\mathrm{T}}(N), \end{aligned} \tag{10.74}$$

care este de fapt (10.56) din Tabelul 10.2.

Observaţii. 1. Deşi s-a urmărit obţinerea unor relaţii de recurenţă în ordin, unele din ecuaţiile din Tabelul 10.2. sunt "impure", deoarece implică şi o recurenţă în timp. Recurenţa încrucişată ordin–timp nu constituie o problemă în cazul în care mărimea căreia i se modifică argumentul temporal este disponibilă la toate momentele de timp anterioare. Pentru rezolvarea problemei obţinerii predictorilor doar la anumite momente de timp vor fi necesare totuşi recursii pure în ordin, ceea ce se va realiza prin combinarea relaţiilor de tipul (10.58) cu relaţii recurente invers în timp, pentru a schimba argumentul $N-1$ în argument N.

2. Pentru mărimile auxiliare $\alpha_n(N)$ şi $\alpha_n^y(N)$ nu se pot obţine relaţii recurente în ordin. Calculul rapid al acestor mărimi va fi posibil prin utilizarea unor recursii în timp.

Scanned with OKEN Scanner

Ecuaţie de bază	Inmulţire la stânga	Inmulţire la dreapta	Ecuaţie rezultată
$R_{n+1}^{-1}(N)$ (10.26)		$r_{n+1}(N)$ (10.20)	(10.56)
$R_{n+1}^{-1}(N)$ (10.26)		$r_{n+1}^{f}(N)$ (10.47)	(10.57)
$R_{n+1}^{-1}(N)$ (10.27)		$r_{n+1}^{b}(N)$ (10.48)	(10.58)
$R_{n+1}^{-1}(N)$ (10.26)		$\varphi_{n+1}(N)$ (10.18)	(10.59)
$R_{n+1}^{-1}(N)$ (10.27)		$\varphi_{n+1}(N)$ (10.19)	(10.60)
$R_{n+1}^{-1}(N-1)$ (10.26)		$\varphi_{n+1}(N)$ (10.18)	(10.61)
$R_{n+1}^{-1}(N-1)$ (10.27)		$\varphi_{n+1}(N)$ (10.19)	(10.62)
$R_{n+1}^{-1}(N)$ (10.26)	$r_{n+1}^{\mathrm{T}}(N)$ (10.20)	$r_{n+1}(N)$ (10.20)	(10.63)
$R_{n+1}^{-1}(N)$ (10.26)	$r_{n+1}^{f\mathrm{T}}(N)$ (10.47)	$r_{n+1}^{f}(N)$ (10.47)	(10.64)
$R_{n+1}^{-1}(N)$ (10.27)	$r_{n+1}^{b\mathrm{T}}(N)$ (10.48)	$r_{n+1}^{b}(N)$ (10.48)	(10.65)
$R_{n+1}^{-1}(N)$ (10.26)	$r_{n+1}^{\mathrm{T}}(N)$ (10.20)	$\varphi_{n+1}(N)$ (10.18)	(10.66)
$R_{n+1}^{-1}(N)$ (10.26)	$r_{n+1}^{f\mathrm{T}}(N)$ (10.47)	$\varphi_{n+1}(N)$ (10.18)	(10.67)
$R_{n+1}^{-1}(N)$ (10.27)	$r_{n+1}^{b\mathrm{T}}(N)$ (10.48)	$\varphi_{n+1}(N)$ (10.19)	(10.68)
$R_{n+1}^{-1}(N-1)$ (10.26)	$r_{n+1}^{\mathrm{T}}(N)$ (10.20)	$\varphi_{n+1}(N)$ (10.18)	(10.69)
$R_{n+1}^{-1}(N-1)$ (10.26)	$r_{n+1}^{f\mathrm{T}}(N-1)$ (10.47)	$\varphi_{n+1}(N)$ (10.18)	(10.70)
$R_{n+1}^{-1}(N-1)$ (10.27)	$r_{n+1}^{b\mathrm{T}}(N-1)$ (10.48)	$\varphi_{n+1}(N)$ (10.19)	(10.71)
$R_{n+1}^{-1}(N)$ (10.26)	$\varphi_{n+1}^{\mathrm{T}}(N)$ (10.18)	$\varphi_{n+1}(N)$ (10.18)	(10.72)
$R_{n+1}^{-1}(N)$ (10.27)	$\varphi_{n+1}^{\mathrm{T}}(N)$ (10.19)	$\varphi_{n+1}(N)$ (10.19)	(10.73)

Tabelul 10.1. Deducerea ecuaţiilor recursive în ordin.

Relaţii recurente în timp

Din relaţiile de definiţie (10.13),(10.14),(10.23) şi (10.24) rezultă imediat următoarele relaţii de recurenţă în timp:

$$R_n(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)\varphi_n^{\mathrm{T}}(t) = \lambda R_n(N-1) + \varphi_n(N)\varphi_n^{\mathrm{T}}(N), \tag{10.75}$$

$$r_n(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)y^{\mathrm{T}}(t) = \lambda r_n(N-1) + \varphi_n(N)y^{\mathrm{T}}(N), \tag{10.76}$$

$$r_n^f(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)u^{\mathrm{T}}(t) = \lambda r_n^f(N-1) + \varphi_n(N)u^{\mathrm{T}}(N), \tag{10.77}$$

$$r_n^b(N) = \sum_{t=0}^{N} \lambda^{N-t}\varphi_n(t)u^{\mathrm{T}}(t-n-1) = \lambda r_n^b(N) + \varphi_n(N)u^{\mathrm{T}}(N-n-1). \tag{10.78}$$

Relaţia (10.75) este echivalentă cu factorizarea

$$\begin{bmatrix} R_n(N) & \varphi_n(N) \\ \varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} = \begin{bmatrix} I & \varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda R_n(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix}, \tag{10.79}$$

care prin inversarea matricelor implicate va conduce la relaţii exprimând recurenţa în timp a matricei $R_n^{-1}(N)$.

Scanned with OKEN Scanner

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \alpha_n^{yT}(N) \tag{10.56}$$

$$\theta_{n+1}^f(N) = \begin{bmatrix} \theta_n^f(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \alpha_n(N) \tag{10.57}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N-1) \end{bmatrix} + \begin{bmatrix} I \\ -\theta_n^f(N-1) \end{bmatrix} (\sigma_n^f(N-1))^{-1} \alpha_n^{\mathrm{T}}(N-1) \tag{10.58}$$

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1} \varepsilon_n^b(N) \tag{10.59}$$

$$K_{n+1}^*(N) = \begin{bmatrix} 0 \\ K_n^*(N-1) \end{bmatrix} + \begin{bmatrix} I \\ -\theta_n^f(N-1) \end{bmatrix} (\sigma_n^f(N-1))^{-1} \varepsilon_n^f(N-1) \tag{10.60}$$

$$K_{n+1}(N) = \begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} + \frac{1}{\lambda} \begin{bmatrix} -\theta_n^b(N-1) \\ I \end{bmatrix} (\sigma_n^b(N-1))^{-1} e_n^b(N) \tag{10.61}$$

$$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} + \frac{1}{\lambda} \begin{bmatrix} I \\ -\theta_n^f(N-2) \end{bmatrix} (\sigma_n^f(N-2))^{-1} e_n^f(N-1) \tag{10.62}$$

$$\sigma_{n+1}(N) = \sigma_n(N) - \alpha_n^y(N)(\sigma_n^b(N))^{-1}\alpha_n^{yT}(N) \tag{10.63}$$

$$\sigma_{n+1}^f(N) = \sigma_n^f(N) - \alpha_n^{\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\alpha_n(N) \tag{10.64}$$

$$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \alpha_n(N-1)(\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1) \tag{10.65}$$

$$\varepsilon_{n+1}(N) = \varepsilon_n(N) - \alpha_n^y(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) \tag{10.66}$$

$$\varepsilon_{n+1}^f(N) = \varepsilon_n^f(N) - \alpha_n^{\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) \tag{10.67}$$

$$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) - \alpha_n(N-1)(\sigma_n^f(N-1))^{-1}\varepsilon_n^f(N-1) \tag{10.68}$$

$$e_{n+1}(N) = e_n(N) - \alpha_n^y(N-1)(\sigma_n^b(N-1))^{-1}e_n^b(N) \tag{10.69}$$

$$e_{n+1}^f(N) = e_n^f(N) - \alpha_n^{\mathrm{T}}(N-1)(\sigma_n^b(N-1))^{-1}e_n^b(N) \tag{10.70}$$

$$e_{n+1}^b(N) = e_n^b(N-1) - \alpha_n(N-2)(\sigma_n^f(N-2))^{-1}e_n^f(N-1) \tag{10.71}$$

$$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \varepsilon_n^{bT}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) \tag{10.72}$$

$$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \varepsilon_n^{fT}(N-1)(\sigma_n^f(N-1))^{-1}\varepsilon_n^f(N-1) \tag{10.73}$$

Tabelul 10.2. Ecuaţii recursive în ordin.

Scanned with OKEN Scanner

Alpicând lema de inversare a matricelor partiţionate membrului stâng al relaţiei (10.79) rezultă

$$\begin{bmatrix} R_n(N) & \varphi_n(N) \\ \varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix}^{-1} =$$

$$= \begin{bmatrix} R_n^{-1}(N) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -R_n^{-1}(N)\varphi_n(N) \\ 1 \end{bmatrix} \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix} \frac{1}{1-\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N)\varphi_n(N)} =$$

$$= \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \begin{bmatrix} -K_n^{*\mathrm{T}}(N) & 1 \end{bmatrix} \frac{1}{\gamma_n^0(N)}. \tag{10.80}$$

Utilizând factorizarea (10.79) rezultă imediat

$$\begin{bmatrix} R_n(N) & \varphi_n(N) \\ \varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix}^{-1} = \begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix}. \tag{10.81}$$

Egalând expresiile din (10.80) şi (10.81), rezultă ecuaţia de bază pentru deducerea tuturor ecuaţiilor recursive în timp

$$\begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} +$$

$$+ \begin{bmatrix} -R_n^{-1}(N)\varphi_n(N) \\ 1 \end{bmatrix} \begin{bmatrix} -\varphi_n(N)R_n^{-1}(N) & 1 \end{bmatrix} \frac{1}{\gamma_n^0(N)}. \tag{10.82}$$

Ultima matrice din (10.81) vor realiza trecerea de la argumentul N la $N-1$ pentru mărimile $r_n(N), r_n^f(N)$ şi $r_n^b(N)$:

$$\begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} r_n(N) - \varphi_n(N)y^{\mathrm{T}}(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} \lambda r_n(N-1) \\ y^{\mathrm{T}}(N) \end{bmatrix} \tag{10.83}$$

$$\begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n^f(N) \\ u^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} r_n^f(N) - \varphi_n(N)u^{\mathrm{T}}(N) \\ u^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} \lambda r_n^f(N-1) \\ u^{\mathrm{T}}(N) \end{bmatrix} \tag{10.84}$$

$$\begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n^b(N) \\ u^{\mathrm{T}}(N-n-1) \end{bmatrix} = \begin{bmatrix} r_n^b(N) - \varphi_n(N)u^{\mathrm{T}}(N-n-1) \\ u^{\mathrm{T}}(N-n-1) \end{bmatrix} =$$

$$= \begin{bmatrix} \lambda r_n^b(N-1) \\ u^{\mathrm{T}}(N-n-1) \end{bmatrix}. \tag{10.85}$$

Următoarele identităţi:

$$\varepsilon_n^{\mathrm{T}}(N) = \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix}, \tag{10.86}$$

$$\varepsilon_n^{f\mathrm{T}}(N) = \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix} \begin{bmatrix} r_n^f(N) \\ u^{\mathrm{T}}(N) \end{bmatrix}, \tag{10.87}$$

$$\varepsilon_n^{b\mathrm{T}}(N) = \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix} \begin{bmatrix} r_n^b(N) \\ u^{\mathrm{T}}(N-n-1) \end{bmatrix}, \tag{10.88}$$

Scanned with OKEN Scanner

Ecuaţie de bază	Înmulţire la stânga	Înmulţire la dreapta	Ecuaţii rezultate
(10.82)		$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]^{\mathrm{T}}$	(10.89), (10.92)
(10.82)		$\left[r_n^{f\mathrm{T}}(N) \;\; u(N) \right]^{\mathrm{T}}$	(10.90), (10.93)
(10.82)		$\left[r_n^{b\mathrm{T}}(N) \;\; u(N-n-1) \right]^{\mathrm{T}}$	(10.91), (10.94)
(10.82)		$\left[0_n^{\mathrm{T}} \;\; 1 \right]^{\mathrm{T}}$	(10.95), (10.96)
(10.82)	$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]$	$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]^{\mathrm{T}}$	(10.97)
(10.82)	$\left[r_n^{f\mathrm{T}}(N) \;\; u(N) \right]$	$\left[r_n^{f\mathrm{T}}(N) \;\; u(N) \right]^{\mathrm{T}}$	(10.98)
(10.82)	$\left[r_n^{b\mathrm{T}}(N) \;\; u(N-n-1) \right]$	$\left[r_n^{b\mathrm{T}}(N) \;\; u(N-n-1) \right]^{\mathrm{T}}$	(10.99)
(10.82)	$\left[r_n^{\mathrm{T}}(N) \;\; y(N) \right]$	$\left[r_n^{b\mathrm{T}}(N) \;\; u(N-n-1) \right]^{\mathrm{T}}$	(10.100)
(10.82)	$\left[r_n^{b\mathrm{T}}(N) \;\; u(N-n-1) \right]$	$\left[r_n^{f\mathrm{T}}(N) \;\; u(N) \right]^{\mathrm{T}}$	(10.101)

Tabelul 10.3. Deducerea ecuaţiilor recursive în timp.

vor ajuta, împreună cu (10.83)–(10.85), la deducerea prin operaţiile indicate în Tabelul 10.3. a ecuaţiilor recursive în timp prezentate în Tabelul 10.4.

Operaţiile necesare pentru obţinerea recursiei în timp a predictorilor sunt similare, astfel încât în continuare nu va fi detaliată decât deducerea predictorului $\theta_n(N)$, urmând indicaţiile din prima linie a Tabelului 10.3. Inmulţind la dreapta membrul stâng al ecuaţiei (10.82) cu matricea $\left[r_n^{\mathrm{T}}(N) \;\; y^{\mathrm{T}}(N) \right]^{\mathrm{T}}$, rezultă

$$\begin{aligned}
&\begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} \lambda^{-1} R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} = \\
&= \begin{bmatrix} \lambda^{-1} R_n^{-1}(N-1) & 0 \\ -\lambda^{-1} \varphi_n^{\mathrm{T}}(N) R_n^{-1}(N-1) & 1 \end{bmatrix} \begin{bmatrix} \lambda r_n(N-1) \\ y^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} R_n^{-1}(N-1) r_n(N-1) \\ y^{\mathrm{T}}(N) - \varphi_n^{\mathrm{T}}(N) R_n^{-1}(N-1) r_n(N-1) \end{bmatrix} \\
&= \begin{bmatrix} \theta_n(N-1) \\ y^{\mathrm{T}}(N) - \varphi_n^{\mathrm{T}}(N) \theta_n(N-1) \end{bmatrix} = \begin{bmatrix} \theta_n(N-1) \\ e_n^{\mathrm{T}}(N) \end{bmatrix},
\end{aligned} \tag{10.102}$$

iar înmulţind şi membrul drept al ecuaţiei (10.82) cu aceeaşi matrice se obţine

$$\begin{aligned}
&\begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \frac{1}{\gamma_n^0(N)} \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N) R_n^{-1}(N) & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} = \\
&= \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \frac{y^{\mathrm{T}}(N) - \varphi_n^{\mathrm{T}}(N) R_n^{-1}(N) r_n(N)}{\gamma_n^0(N)} = \\
&= \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \frac{\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)}.
\end{aligned} \tag{10.103}$$

Scanned with OKEN Scanner

$$\theta_n(N) = \theta_n(N-1) + K_n^*(N)e_n^{\mathrm{T}}(N) \quad (10.89)$$
$$\theta_n^f(N) = \theta_n^f(N-1) + K_n^*(N)e_n^{f\mathrm{T}}(N) \quad (10.90)$$
$$\theta_n^b(N) = \theta_n^b(N-1) + K_n^*(N)e_n^{b\mathrm{T}}(N) \quad (10.91)$$
$$\varepsilon_n(N) = e_n(N)\gamma_n^0(N) \quad (10.92)$$
$$\varepsilon_n^f(N) = e_n^f(N)\gamma_n^0(N) \quad (10.93)$$
$$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N) \quad (10.94)$$
$$K_n^*(N) = K_n(N)\gamma_n^0(N) \quad (10.95)$$
$$\gamma_n(N) = \frac{1}{\gamma_n^0(N)} \quad (10.96)$$
$$\sigma_n(N) = \lambda\sigma_n(N-1) + \varepsilon_n(N)\varepsilon_n^{\mathrm{T}}(N)/\gamma_n^0(N) \quad (10.97)$$
$$\sigma_n^f(N) = \lambda\sigma_n^f(N-1) + \varepsilon_n^f(N)\varepsilon_n^{f\mathrm{T}}(N)/\gamma_n^0(N) \quad (10.98)$$
$$\sigma_n^b(N) = \lambda\sigma_n^b(N-1) + \varepsilon_n^b(N)\varepsilon_n^{b\mathrm{T}}(N)/\gamma_n^0(N) \quad (10.99)$$
$$\alpha_n^y(N) = \lambda\alpha_n^y(N-1) + \varepsilon_n(N)\varepsilon_n^{b\mathrm{T}}(N)/\gamma_n^0(N) \quad (10.100)$$
$$\alpha_n(N) = \lambda\alpha_n(N-1) + \varepsilon_n^b(N)\varepsilon_n^{f\mathrm{T}}(N)/\gamma_n^0(N) \quad (10.101)$$

Tabelul 10.4. Relaţii de bază pentru recurenţa în timp.

Egalând expresiile (10.50) şi (10.51) rezultă

$$\begin{bmatrix} \theta_n(N-1) \\ e_n^{\mathrm{T}}(N) \end{bmatrix} = \begin{bmatrix} \theta_n(N) - K_n^*(N)\frac{\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)} \\ \frac{\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)} \end{bmatrix}, \quad (10.104)$$

iar această relaţie este echivalentă cu relaţiile (10.89) şi (10.92). Având în vedere structura similară a relaţiilor (10.83)–(10.85) şi a relaţiilor (10.86)–(10.88), vor rezulta imediat recursiile (10.90),(10.91) şi relaţiile (10.93),(10.94).

Operaţiile indicate în linia a 4-a a Tabelului 10.3. vor transforma membrul stâng al relaţiei (10.82) astfel

$$\begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0_n \\ 1 \end{bmatrix} =$$
$$= \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ -\lambda^{-1}\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1) & 1 \end{bmatrix} \begin{bmatrix} -\varphi_n(N) \\ 1 \end{bmatrix} =$$
$$= \begin{bmatrix} -\lambda^{-1}R_n^{-1}(N-1)\varphi_n(N) \\ 1 + \lambda^{-1}\varphi_n^{\mathrm{T}}(N)R^{-1}(N-1)\varphi_n(N) \end{bmatrix} = \begin{bmatrix} -K_n(N) \\ \gamma_n(N) \end{bmatrix}. \quad (10.105)$$

Mărimea notată

$$\gamma_n(N) = 1 + \lambda^{-1}\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1)\varphi_n(N) \quad (10.106)$$

are semnificaţia raportului apriori/aposteriori, aşa cum rezultă înmulţind membrul drept al

Scanned with OKEN Scanner

ecuaţiei (10.82) cu $\left[\, 0_n^{\mathrm{T}} \;\; 1 \,\right]^{\mathrm{T}}$

$$\left\{\begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \frac{1}{\gamma_n^0(N)} \begin{bmatrix} -K_n^{*\mathrm{T}}(N) & 1 \end{bmatrix}\right\} \begin{bmatrix} 0_n \\ 1 \end{bmatrix} =$$
$$= \begin{bmatrix} -K_n^*(N) \\ 1 \end{bmatrix} \frac{1}{\gamma_n^0(N)} \tag{10.107}$$

şi egalând expresiile din (10.105) şi (10.107) (în acest mod obţinându-se relaţiile (10.95) şi (10.96) din Tabelul 10.4.).

Obţinerea recursiilor în timp ale matricelor de covarianţă $\sigma_n(N), \sigma_n^f(N)$ şi $\sigma_n^b(N)$ ca şi ale matricelor de intercovarianţă $\alpha_n^y(N)$ şi $\alpha_n(N)$ implică acelaşi tip de operaţii, în continuare detaliindu-se numai operaţiile din linia a 5-a a Tabelului 10.3.

Înmulţind membrul stâng al relaţiei (10.82) la stânga şi la dreapta cu matricele indicate în Tabelul 10.3. se obţine

$$\begin{bmatrix} r_n^{\mathrm{T}}(N) & y(N) \end{bmatrix} \begin{bmatrix} I & 0 \\ -\varphi_n^{\mathrm{T}}(N) & 1 \end{bmatrix} \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\varphi_n(N) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} =$$
$$= \begin{bmatrix} \lambda r_n^{\mathrm{T}}(N-1) & y(N) \end{bmatrix} \begin{bmatrix} \lambda^{-1}R_n^{-1}(N-1) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda r_n(N-1) \\ y^{\mathrm{T}}(N) \end{bmatrix} =$$
$$= \lambda r_n^{\mathrm{T}}(N-1)R_n^{-1}(N-1)r_n(N-1) + y(N)y^{\mathrm{T}}(N). \tag{10.108}$$

Înmulţind cu aceleaşi matrice şi membrul drept din (10.82) rezultă

$$\begin{bmatrix} r_n^{\mathrm{T}}(N) & y(N) \end{bmatrix} \left\{\begin{bmatrix} I \\ 0 \end{bmatrix} R_n^{-1}(N) \begin{bmatrix} I & 0 \end{bmatrix} + \begin{bmatrix} -R_n^{-1}(N)\varphi_n(N) \\ 1 \end{bmatrix} \begin{bmatrix} -\varphi_n^{\mathrm{T}}(N)R_n^{-1}(N) & 1 \end{bmatrix} \cdot \right.$$
$$\left. \cdot \frac{1}{\gamma_n^0(N)} \right\} \begin{bmatrix} r_n(N) \\ y^{\mathrm{T}}(N) \end{bmatrix} = r_n^{\mathrm{T}}(N)R_n^{-1}(N)r_n(N) + \frac{\varepsilon_n(N)\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)}. \tag{10.109}$$

Egalăm expresiile din (10.108) şi (10.109) şi scădem din ambii membri $\sum_{t=0}^{N-1} \lambda^{N-t} y(t)y^{\mathrm{T}}(t)$, rezultând

$$\lambda\left(r_n^{\mathrm{T}}(N-1)R_n^{-1}(N-1)r_n(N-1) - \sum_{t=0}^{N-1} \lambda^{N-t-1} y(t)y^{\mathrm{T}}(t)\right) + y(N)y^{\mathrm{T}}(N) =$$
$$= r_n^{\mathrm{T}}(N)R_n^{-1}(N)r_n(N) - \sum_{t=0}^{N-1} \lambda^{N-t} y(t)y^{\mathrm{T}}(t) + \frac{\varepsilon_n(N)\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)}, \tag{10.110}$$

sau

$$-\lambda\sigma_n(N-1) = -\sigma_n(N) + \frac{\varepsilon_n(N)\varepsilon_n^{\mathrm{T}}(N)}{\gamma_n^0(N)}, \tag{10.111}$$

de unde rezultă imediat (10.97).

Datorită structurii foarte asemănătoare a matricelor de covarianţă, relaţiile (10.98)–(10.101) vor rezulta similar cu (10.97).

Scanned with OKEN Scanner

Algoritmi latice pentru erorile de predicţie

În unele aplicaţii obţinerea vectorului $\theta_n(N)$ al parametrilor modelului nu reprezintă un scop în sine, ci un mijloc pentru calculul predicţiilor

$$\hat{y}_n(N) = \theta_n^{\mathrm{T}}(N)\varphi_n(N) \tag{10.112}$$

pentru valorile $1 \leq n \leq n_{max}$ şi pentru calculul matricelor de covarianţă $\{\sigma_n(N)\}_{n=1}^{n_{max}}$ ale erorilor de predicţie, de exemplu pentru a fi utilizate în proceduri de alegere a ordinului modelului.

Observăm din (10.39) că o relaţie mai rapidă decât (10.112) pentru calculul predicţiilor este

$$\hat{y}_n(N) = y(N) - \varepsilon_n(N) \tag{10.113}$$

şi în consecinţă, vom considera în continuare problema evaluării rapide a mărimilor $\{\varepsilon_n(N)\}_{n=1}^{n_{max}}$ şi $\{\sigma_n(N)\}_{n=1}^{n_{max}}$.

Examinând relaţiile de recurenţă disponibile în Tabelele 10.2. şi 10.4. rezultă mai multe variante de selectare a unor seturi complete de ecuaţii recursive pentru rezolvarea acestei probleme.

Într-o primă variantă sunt utilizate relaţii recursive în ordin pentru mărimile:

- erorile de predicţie aposteriori $\varepsilon_n(N), \varepsilon_n^f(N), \varepsilon_n^b(N)$ din (10.66)–(10.68);
- matricele de covarianţă $\sigma_n(N), \sigma_n^f(N), \sigma_n^b(N)$ din (10.63)–(10.65);
- raportul aposteriori/apriori $\gamma_n^0(N)$ din (10.72)

şi relaţii recursive în timp pentru propagarea matricelor de intercovarianţă $\alpha_n^y(N)$ şi $\alpha_n(N)$, (10.100),(10.101).

Algoritmul care rezultă este prezentat în Tabelul 10.5. Valorile de iniţializare a recursiilor în ordin (pentru ordinul fictiv $n = 0$) rezultă din relaţiile de definiţie a mărimilor implicate (considerând formal $\theta_0 = 0$). Iniţializarea recursiilor în timp (la momentul $N = 0$) cu valori nule pentru toate mărimile recurente în timp concordă cu ipoteza "fereastră anterioară" (10.11), dar conduce la următoarea situaţie: cel puţin pe durata a câtorva primi paşi, matricele σ_n^f şi σ_n^b vor fi singulare, iar recursiile algoritmului utilizează inversele acestor matrice. Se poate arăta[66] că ecuaţiile recursive rămân valide dacă sunt utilizate pseudoinverse în locul inverselor matricelor σ_n^f şi σ_n^b (în cazul scalar, datorită faptului că σ_n^f şi σ_n^b corespund secţiunii de modelare AR a semnalului $u(t)$, tratarea singularităţii se face ca în Secţiunea 7.3).

O altă variantă de algoritm latice utilizează relaţii recursive în ordin pentru

- erorile de predicţie aposteriori şi apriori, relaţiile (10.66)–(10.71);
- matricele de covarianţă, relaţiile (10.63)–(10.65);

şi relaţii recursive în timp, echivalente cu (10.100),(10.101):

$$\alpha_n^y(N) = \lambda\alpha_n^y(N-1) + e_n(N)\varepsilon_n^{b\mathrm{T}}(N), \tag{10.114}$$

$$\alpha_n(N) = \lambda\alpha_n(N-1) + e_n^b(N)\varepsilon_n^{f\mathrm{T}}(N), \tag{10.115}$$

iar valorile de iniţializare în ordin şi timp sunt similare cu cele din Algoritmul 10.1.

Există şi alte variante de algoritmi, dar toate sunt echivalente din punctul de vedere al numărului de operaţii necesar, care este de ordinul $\mathcal{O}(n_{max}q^3)$.

Scanned with OKEN Scanner

Algoritmul 10.1.1.

ALGORITM LATICE PENTRU ERORILE DE PREDICŢIE		*Număr operaţii/ ecuaţie* ⋆ / +	*Număr operaţii/ pas alg.*
1. *Iniţializare la momentul* $(N = 0)$			
Pentru $n = 0, \ldots, n_{max} - 1$			
$\alpha_n^y(0) = 0; \quad \alpha_n(0) = 0$			
$\sigma_0(0) = 0$			
$\sigma_0^f(0) = \sigma_0^b(0) = 0$			
2. *Recursii în timp*			$\mathcal{O}(n_{max}^2)$
Pentru $N = 1, 2, \ldots, N_{max}$			
2.1 *Iniţializare recursii în ordin*			
$\varepsilon_0(N) = y(N); \quad \varepsilon_0^f(N) = u(N); \quad \varepsilon_0^b(N) = u(N-1)$			
$\gamma_0^0(N) = 1$			
$\sigma_0(N) = \lambda\sigma_0(N-1) + y(N)y^{\mathrm{T}}(N)$	(1)	2 0 1	
$\sigma_0^f(N) = \lambda\sigma_0^f(N-1) + u(N)u^{\mathrm{T}}(N)$	(2)	2 0 1	
$\sigma_0^b(N) = \sigma_0^f(N-1)$			
2.2 *Recursii în ordin:*			$\mathcal{O}(n_{max})$
Pentru $n = 0, \ldots, n_{max} - 1$			
$\alpha_n^y(N) = \lambda\alpha_n^y(N-1) + \varepsilon_n(N)\varepsilon_n^{b\mathrm{T}}(N)/\gamma_n^0(N)$	(3)	2 1 1	
$\alpha_n(N) = \lambda\alpha_n(N-1) + \varepsilon_n^b(N)\varepsilon_n^{f\mathrm{T}}(N)/\gamma_n^0(N)$	(4)	2 1 1	
$\varepsilon_{n+1}(N) = \varepsilon_n(N) - \alpha_n^y(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)$	(5)	1 1 1	
$\varepsilon_{n+1}^f(N) = \varepsilon_n^f(N) - \alpha_n^{\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)$	(6)	1 1 1	
$\varepsilon_{n+1}^b(N) = \varepsilon_n^b(N-1) - \alpha_n(N-1)(\sigma_n^f(N-1))^{-1}\varepsilon_n^f(N-1)$	(7)	1 1 1	
$\sigma_{n+1}(N) = \sigma_n(N) - \alpha_n^y(N)(\sigma_n^b(N))^{-1}\alpha_n^{y\mathrm{T}}(N)$	(8)	1 1 1	
$\sigma_{n+1}^f(N) = \sigma_n^f(N) - \alpha_n^{\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\alpha_n(N)$	(9)	1 1 1	
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \alpha_n(N-1)(\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1)$	(10)	1 1 1	
$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \varepsilon_n^{b\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)$	(11)	1 1 1	
$\hat{y}_{n+1}(N) = y(N) - \varepsilon_{n+1}(N)$			
$(\star)$: *Dacă matricea* σ *este singulară,* σ^{-1} *denotă o pseudoinversă*			
Total (cazul scalar): $11n_{max}$ *înmulţiri;* $9n_{max}$ *împărţiri;* $9n_{max}$ *adunări*			

Tabelul 10.5. Algoritmul "latice a erorilor de predicţie" pentru modele multivariabile de tip sumă de convoluţie.

Scanned with OKEN Scanner

Parametrizarea latice a modelelor de tip sumă de convoluţie

Pentru simplitate, în continuare se vor face referiri numai la cazul scalar.

Grupurile de ecuaţii (10.66)–(10.68), (10.69)–(10.71) evidenţiază structuri modulare similare cu cea a laticii descrise în Secţiunea 7.2 pentru parametrizarea modelelor AR. În structurile latice asociată ecuaţiilor (10.66)–(10.68) trebuie utilizate următoarele definiţii ale coeficienţilor de reflexie:

$$K_n^b(N) = -\alpha_n(N-1)(\sigma_n^f(N-1))^{-1}, \tag{10.116}$$

$$K_n^f(N) = -\alpha_n(N)(\sigma_n^b(N))^{-1}, \tag{10.117}$$

$$K_n^y(N) = -\alpha_n^y(N)(\sigma_n^b(N))^{-1}. \tag{10.118}$$

Setul de coeficienţi de reflexie $\{K_n^b, K_n^f, K_n^y\}_{n=1}^{n_{max}}$ constituie o altă parametrizare a modelelor de tip sumă de convoluţie, cu un număr de parametri mult mai mic decât parametrizarea directă $\{\theta_n, \theta_n^f, \theta_n^b\}_{n=1}^{n_{max}}$.

Rezultă cu claritate tehnica prin care se produce această parametrizare: se modelează autoregresiv semnalul $u(t)$, iar erorile $\varepsilon_n(t)$ se obţin printr-o modalitate asemănătoare realizării directe din figura 7.1.*a*, ca o combinaţie liniară între $y(t)$ şi erorile de predicţie înapoi de diferite ordine.

Trecerea de la parametrizarea latice la parametrizarea directă se poate obţine în cazul staţionar prin recursiile de tip Levinson (10.56)–(10.58) (în care datorită staţionarităţii se omite argumentul dependenţei de timp, N), iar în cazul nestaţionar printr-o modificare a recursiilor (10.56)–(10.58), prezentată în secţiunea următoare.

10.2. Algoritmul "Latice a predictorilor"

În anumite aplicaţii ale modelării (de exemplu conducerea adaptivă a sistemelor multivariable), facilităţile oferite de secţiunea latice a erorilor de predicţie (calculul predicţiilor, determinarea ordinului) nu sunt suficiente, fiind cerute în mod explicit şi matricele parametrilor $\theta_n(N)$. De multe ori însă parametrii sunt ceruţi cu o frecvenţă mult mai redusă decât frecvenţa de eşantionare a datelor de intrare-ieşire, aşa încât se impune ca evaluarea predictorilor să fie efectuată utilizând numai datele disponibile în secţiunea latice (care este cel mai rapid mijloc de propagare în timp a informaţiei necesare). Relaţiile (10.56)–(10.58) din Tabelul 10.2 rezolvă parţial această problemă, determinând predictorul $\theta_n(N)$ în mod recursiv în ordin plecând de la coeficienţii de reflexie, dar pentru realizarea recursiei (10.58) sunt necesari şi predictorii la pasul anterior de timp. Recursiile (10.90) şi (10.91) din Tabelul 10.3. pot realiza trecerea argumentelor de la valoarea $N-1$ la valoarea N, relaţia (10.58) rescriindu-se (utilizând (10.93) şi (10.94))

$$\begin{aligned}
\theta_{n+1}^b(N) &= \begin{bmatrix} 0 \\ \theta_n^b(N) - K_n^*(N)e_n^{b\mathrm{T}}(N) \end{bmatrix} + \\
&+ \begin{bmatrix} I \\ -\theta_n^f(N) + K_n^*(N)e_n^{f\mathrm{T}}(N) \end{bmatrix} (\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1) = \\
&= \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} + \begin{bmatrix} I \\ -\theta_n^f(N) \end{bmatrix} (\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1) - \\
&- \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix} \frac{\varepsilon_n^{b\mathrm{T}}(N) - (\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1)\varepsilon_n^{f\mathrm{T}}(N)}{\gamma_n^0(N)}.
\end{aligned} \tag{10.119}$$

Din (10.119) rezultă că în cadrul recursiilor în ordin trebuie inclusă şi relaţia (10.59) de calcul a amplificărilor Kalman. Secţiunea opţională de calcul a predictorilor este prezentată în Tabelul 10.6. Iniţializarea la ordinul $n = 1$ a predictorilor rezultă imediat din definiţii.

De remarcat că singularitatea matricelor σ_n^b sau σ_n^f este echivalentă cu singularitatea matricei R_{n+1} şi în acest caz predictorii de ordin $n+1$ nu mai sunt unici. Se poate demonstra [65] că înlocuirea inverselor matricelor σ_n^b şi σ_n^f prin pseudoinverse este echivalentă cu înlocuirea în definiţiile (10.16),(10.17), (10.28)-(10.31) a inversei R_n^{-1} cu o pseudoinversă, deci predictorii furnizaţi de algoritm rămân în continuare soluţii (neunice) ale problemei CMMP.

De remarcat un caz particular important în aplicaţii al modelelor multivariabile de tip sumă de convoluţie şi anume modelul multivariabil autoregresiv, care se obţine pentru alegerea (10.5) a intrărilor în model. În acest caz, toate mărimile care au indice superior "f" sunt identice cu cele fără indice superior (e.g. $\theta_n^f(N) \equiv \theta_n(N)$), iar $\alpha_n^y(N) \equiv \alpha_n^{\mathrm{T}}(N)$. În consecinţă, din Algoritmul 10.1.1 vor dispare ecuaţiile (1),(3),(6),(9),(11),(15). Se poate observa că pentru cazul scalar, algoritmul astfel rezultat coincide cu Algoritmul 7.4 (Tabelul 7.8.) cu excepţia semnului predictorilor care va trebui inversat, datorită diferenţei de semn în parametrizările iniţiale (1.1) şi (10.6).

Extinderea Algoritmului 10.1.1.

SECŢIUNEA OPŢIONALĂ PENTRU CALCULUL PREDICTORILOR

2. Recursii în timp

2.3 Dacă la pasul N se doreşte evaluarea predictorilor

Atunci:

2.3.1 Iniţializare recursii în ordin:

$$\theta_1(N) = (\sigma_0^b(N))^{-1}\alpha_0^{y\mathrm{T}}(N) \tag{12}$$

$$\theta_1^f(N) = (\sigma_0^b(N))^{-1}\alpha_0(N) \tag{13}$$

$$\theta_1^b(N) = (\sigma_0^f(N-1))^{-1}\alpha_0^{\mathrm{T}}(N-1) \tag{14}$$

$$K_1^*(N) = (\sigma_0^b(N))^{-1}\varepsilon_0^b(N) \tag{15}$$

2.3.2 Recursii în ordin:

Pentru $n = 1, \ldots, n_{max} - 1$

$$\theta_{n+1}(N) = \begin{bmatrix} \theta_n(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1}\alpha_n^{y\mathrm{T}}(N) \tag{16}$$

$$\theta_{n+1}^f(N) = \begin{bmatrix} \theta_n^f(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1}\alpha_n(N) \tag{17}$$

$$\theta_{n+1}^b(N) = \begin{bmatrix} 0 \\ \theta_n^b(N) \end{bmatrix} + \begin{bmatrix} I \\ -\theta_n^f(N) \end{bmatrix} (\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1) -$$

$$- \begin{bmatrix} 0 \\ K_n^*(N) \end{bmatrix} (\varepsilon_n^{b\mathrm{T}}(N) - \varepsilon_n^{f\mathrm{T}}(N)(\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1))/\gamma_n^0(N) \tag{18}$$

$$K_{n+1}^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta_n^b(N) \\ I \end{bmatrix} (\sigma_n^b(N))^{-1}\varepsilon_n^b(N) \tag{19}$$

Total operaţii (în cazul scalar): $\mathcal{O}(n_{max}^2)$ flops

Tabelul 10.6. Extinderea Algoritmului 10.1.1 cu o secţiune pentru trecerea de la erorile de predicţie la predictori.

Scanned with OKEN Scanner

Scanned with OKEN Scanner

Capitolul 11

Algoritmul CMMP cu Propagarea Covarianţelor pentru Modele Tip Sumă de Convoluţie

11.1. Algoritmul CMMP cu propagarea covarianţelor

Estimarea parametrilor este necesară, în foarte multe aplicaţii, numai la intervale de timp mult mai mari decât perioada de eşantionare a intrării şi ieşirii modelului. Aplicarea unor metode off-line de estimare implică memorarea întregii evoluţii a intrărilor şi ieşirilor şi prelucrarea acestui set de date atunci când sunt necesare estimaţiile parametrilor. Această soluţie este foarte ineficientă atunci când numărul de eşantioane este mare. Prin utilizarea unor metode de estimare recursive în timp, la fiecare pas de eşantionare se realizează actualizarea unor mărimi intermediare, mărimi din care se poate deduce estimaţia parametrilor, la orice moment de timp. Un astfel de tip de procedură (numită bloc–recursivă) a fost utilizată în Secţiunea 7.1 pentru modelarea AR. Acolo, covarianţele de eşantion au fost alese ca mărimi intermediare pentru propagarea informaţiei în timp, iar la momentele de timp când se doreşte obţinerea parametrilor AR, se utilizează algoritmul Levinson–Durbin.

Datorită simplităţii atât a relaţiilor de actualizare a covarianţelor (7.14)(7.15) cât şi a algoritmului de estimare, această metodă este foarte eficientă numeric, necesitând n_{max} adunări şi înmulţiri la fiecare pas de eşantionare pentru propagarea covarianţelor şi $\mathcal{O}(n_{max}^2)$ operaţii pentru evaluarea estimaţiilor.

Îmbunătăţirea calităţii estimaţiilor pentru segmente de date de lungimi mici s-a obţinut prin minimizarea exactă a unui criteriu CMMP determinist, procedura bloc-recursivă prezentată în Secţiunile 7.2 ,7.3 necesitând aproximativ $8n_{max}$ operaţii la fiecare pas de eşantionare pentru propagarea mărimilor recursive în timp (care de fapt definesc parametrizarea latice a modelelor AR) şi $\mathcal{O}(n_{max}^2)$ operaţii pentru estimarea parametrilor modelului AR (definiţi prin parametrizarea directă (1.1),(1.4)).

În continuare va fi prezentată o altă procedură bloc recursivă pentru problema minimizării criteriului CMMP determinist, care utilizează ca mărimi de propagare în timp a informaţiei covarianţele de eşantion. În acest mod se va reduce foarte mult efortul de calcul necesar la fiecare pas de eşantionare (aproximativ $2n_{max}$ operaţii, faţă de $8n_{max}$ operaţii la algoritmii latice) dar în schimb va creşte numărul de operaţii necesar pentru evaluarea estimaţiilor. Noua procedură va furniza aceleaşi estimaţii ca şi algoritmul latice, la un cost mediu (număr de

operaţii mediu/ pas de eşantionare) cu atât mai bun faţă de procedura latice, cu cât intervalul de timp dintre două evaluări ale estimaţiilor parametrilor este mai mare. Metoda descrisă în acest capitol poate fi considerată ca o modificare a procedurii Levinson–Durbin de trecere de la covarianţele de eşantion la parametrii modelului, astfel încât să fie rezolvat sistemul (7.34)–(7.38) pentru minimizarea criteriului determinist (7.32), în locul rezolvării sistemului (1.13) pentru minimizarea criteriului statistic (1.6).

Vom considera problema estimării parametrilor tuturor modelelor multivariabile de tip sumă de convoluţie, de ordine de la 1 la n_{max}, pentru că aşa cum s-a văzut în Secţiunea 10.1 aceste modele sunt foarte generale, prin particularizare obţinându-se multe alte clase de modele des folosite în practică, astfel încât algoritmul rapid obţinut se va putea aplica cu mici modificări şi pentru alte clase de modele.

În Capitolul 10 a fost analizată sistematic recursivitatea în ordin şi timp a soluţiei $\theta_n(N)$ a problemei de modelare multivariabilă prin modele de tip sumă de convoluţie; relaţiile rezultate, prezentate în Tabelele 10.2., 10.4., vor constitui de fapt "piesele" prin a căror combinare se va obţine procedura bloc recursivă cu propagarea covarianţelor.

În continuare vom presupune adevărată ipoteza "fereastră anterioară" (10.11) şi vom utiliza notaţiile din Capitolul 10. Deoarece în Secţiunea 11.2 vor fi considerate în mod detaliat diferitele tehnici de deponderare a informaţiei "vechi", în acest paragraf se va lua $\lambda = 1$.

Soluţia care minimizează criteriul matriceal (10.12) este dată de sistemul (10.16), definit la momentul N prin valoarea $r_n(N)$ şi prin matricea $R_n(N)$. Elementele matricei $R_n(N)$ sunt matricele de autocorelaţie (10.25):

$$\{\rho_0^u(t)\}_{t=N-n}^{N-1},\ \{\rho_1^u(t)\}_{t=N-n+1}^{N-1},\ \ldots,\rho_{n-1}^u(N-1). \tag{11.1}$$

Datorită structurii "aproape Toeplitz" a matricei $R_n(N)$, cele $(n^2+n)/2$ elemente din (11.1) sunt complet determinate dacă se cunosc prima linie a matricei $R_n(N)$:

$$\begin{aligned}\sum_{t=0}^{N} u(t-1)\varphi_n^{\mathrm{T}}(t) &= \left[\ \rho_0^u(N-1)\ \ \rho_1^{u\mathrm{T}}(N-1)\ \ \ldots\ \ \rho_{n-1}^{u\mathrm{T}}(N-1)\ \right] = \\ &= \left[\ \rho_0^u(N-1)\ \ r_{n-1}^{f\mathrm{T}}(N-1)\ \right]\end{aligned} \tag{11.2}$$

şi vectorul

$$\varphi_n(N) = \left[\ u^{\mathrm{T}}(N-1)\ \ \ldots\ \ u^{\mathrm{T}}(N-n)\ \right]^{\mathrm{T}}, \tag{11.3}$$

deoarece un element oarecare $\rho_p^u(N-q)$ din (11.1) $(p < n-1,\ q < n-p)$ se poate calcula utilizând numai (11.2) şi (11.3) astfel:

$$\begin{aligned}\rho_p^u(N-q) &= \sum_{t=0}^{N-q} \lambda^{N-q-t} u(t-p)u^{\mathrm{T}}(t) = \lambda^{-q+1}(\rho_p^u(N-1) - \\ &- u(N-p-1)u^{\mathrm{T}}(N-1) - \ldots - \lambda^{q-2}u(N-p-q+1)u^{\mathrm{T}}(N-q+1)).\end{aligned} \tag{11.4}$$

În consecinţă, prin propagarea în timp a mărimilor

$$\sigma_0^f(N) = \rho_0^u(N),\ \ r_n^f(N),\ \ r_n(N),\ \ \varphi_n(N) \tag{11.5}$$

se asigură posibilitatea calculării în orice moment a estimaţiilor parametrilor. Ecuaţia de adaptare în timp a acestor mărimi rezultă imediat din definiţii:

$$\sigma_0^f(N) = \sum_{t=0}^{N} \lambda^{N-t} u(t)u^{\mathrm{T}}(t) = \lambda\sigma_0^f(N-1) + u(N)u^{\mathrm{T}}(N), \tag{11.6}$$

$$r_n^f(N) = \sum_{t=0}^{N} \lambda^{N-t} \varphi_n(t)u^{\mathrm{T}}(t) = \lambda r_n^f(N-1) + \varphi_n(N)u^{\mathrm{T}}(N), \tag{11.7}$$

$$r_n(N) = \sum_{t=0}^{N} \lambda^{N-t} \varphi_n(t)y^{\mathrm{T}}(t) = \lambda r_n(N-1) + \varphi_n(N)y^{\mathrm{T}}(N). \tag{11.8}$$

Trecerea eficientă, la un moment dat, de la valorile covarianţelor la valorile parametrilor modelului, se va realiza printr-o procedură pur recursivă în ordin, bazată pe ecuaţiile recursive în ordin din Tabelul 10.2. Ecuaţia (10.56) furnizează recursia în ordin pentru predictorul $\theta_{n+1}(N)$, antrenând şi necesitatea evaluării predictorului $\theta_n^b(N)$. La rândul ei, recursia (10.58) a predictorului $\theta_{n+1}^b(N)$ implică necesitatea calculului predictorului $\theta_n^f(N-1)$, care se poate realiza prin (10.57) în care argumentul temporal N este schimbat cu $N-1$. În sfârşit, $\theta_n^b(N-1)$ se poate determina din $\theta_n^b(N)$ prin (10.91) (Tabelul 10.4.) implicând şi iterarea vectorului Kalman aposteriori $K_n^*(N)$ recursiv în ordin prin (10.59).

O parte din celelalte mărimi auxiliare implicate se pot calcula rapid ($\mathcal{O}(q^2)$ operaţii) prin utilizarea relaţiilor simple care le leagă de mărimile de ordin inferior, în acest mod calculându-se $\gamma_{n+1}^0(N), \sigma_{n+1}^f(N-1), \sigma_{n+1}^n(N)$. Atunci când nu există relaţii iterative în ordin, mărimile auxiliare se vor calcula direct din definiţii, dar la un cost numeric mai ridicat ($\mathcal{O}(nq^2)$ operaţii), astfel calculându-se $\varepsilon_n^b(N)$(10.41), $\alpha_n(N-1)$, din (10.43) rescrisă în următorul mod

$$\begin{aligned}\alpha_n(N-1) &= \rho_n^u(N-1) - r_n^{b\mathrm{T}}(N-1)R_n^{-1}(N-1)r_n^f(N-1) = \\ &= \left[\, r_n^f(N-1) \,\right]_n - \theta_n^{b\mathrm{T}}(N-1)r_n^f(N-1)\end{aligned} \tag{11.9}$$

şi $\alpha_n^y(N)$(10.42) rescrisă în funcţie de mărimile care intervin în ecuaţiile reţinute până acum

$$\alpha_n^y(N) = \rho_n^{yu\mathrm{T}}(N) - r_n^{\mathrm{T}}(N)R_n^{-1}(N)r_n^b(N) = \left[\, r_n^{\mathrm{T}}(N) \,\right]_n - r_n^{\mathrm{T}}(N)\theta_n^b(N). \tag{11.10}$$

Pentru a dispune şi de valoarea mărimii $\sigma_n^b(N-1)$, necesară în recursiile (10.57) şi (10.64), vom utiliza relaţia recursivă în timp (10.99) din Tabelul 10.4.

Aceste relaţii au fost ordonate, astfel încât să constituie un algoritm bloc recursiv, în Algoritmul 11.1.1. Deşi nu sunt absolut necesare, în algoritm au fost incluse şi relaţiile notate cu asterix, pentru calculul matricei $\sigma_n(N)$ (necesară în procedurile de determinare a ordinului modelului) şi pentru calculul erorii de predicţie $\varepsilon_n(N)$, furnizând valoarea ieşirii filtrate $\hat{y}_n(N)$.

Relaţiile de iniţializare a recursiilor în ordin sunt foarte asemănătoare cu iniţializarea în ordin utilizată în algoritmul din Tabelul 10.6.

Iniţializarea recursiilor în timp cu valori zero pentru toate mărimile care se propagă în timp corespunde ipotezei "fereastră anterioară" (10.11) considerată în acest capitol.

În cazul scalar, numărul de operaţii implicat în secţiunea recursivă în timp este aproximativ $2n_{max}$, iar numărul de operaţii necesar pentru estimarea setului de parametri optimali $\{\theta_n(N)\}_{n=1}^{n_{max}}$ este de aproximativ $4n_{max}^2$.

Scanned with OKEN Scanner

Algoritmul 11.1.1.

ALGORITM BLOC RECURSIV CU PROPAGAREA COVARIANŢELOR	*Număr operaţii/ ecuaţie*
1. *Iniţializare la momentul* $(N = 0)$	
$r_M(0) = 0;\quad r_M^f(0) = 0;$	
$\sigma_0(0) = 0;\quad \sigma_0^f(0) = \sigma_0^b(0) = 0$	
2. *Recursii în timp*	
Pentru $N = 1, 2, \ldots, N_{max}$	
2.1	
$r_M(N) = r_M(N-1) + \varphi_M(N)y^{\mathrm{T}}(N)$	M
$r_M^f(N) = r_M^f(N-1) + \varphi_M(N)u^{\mathrm{T}}(N)$	M
$\sigma_0(N) = \sigma_0(N-1) + y(N)y^{\mathrm{T}}(N)$	1
$\sigma_0^f(N) = \sigma_0^f(N-1) + u(N)u^{\mathrm{T}}(N)$	1
$\sigma_0^b(N) = \sigma_0^f(N-1)$	
2.2. *Dacă se doreşte evaluarea predictorului*	
Atunci	
2.2.1 *Iniţializare recursii în ordin*	
$\varepsilon_0(N) = y \qquad {}_0^b(N) = u(N-1);\ (N)_0^0(N) = 1$	
$\alpha_0(N-1) = \left[\, r_M^f(N-1) \,\right]_1$	
$\alpha_0^y(N) = \left[\, r_M^{\mathrm{T}}(N) \,\right]_1$	
$\theta_1(N) = (\sigma_0^b(N))^{-1}\alpha_0^{y\mathrm{T}}(N)$	1
$\theta_1^f(N-1) = (\sigma_0^b(N-1))^{-1}\alpha_0(N-1)$	1
$\theta_1^b(N) = (\sigma_0^f(N-1))^{-1}\alpha_0^{\mathrm{T}}(N-1)$	1
$K_1^*(N) = (\sigma_0^b(N))^{-1}\varepsilon_0^b(N)$	1
2.2.2 *Recursii în ordin:*	
Pentru $n = 0, \ldots, M-2$	
$\varepsilon_{n+1}^b(N) = u(N-n-2) - \theta_{n+1}^{b\mathrm{T}}(N)\varphi_{n+1}(N)$	$n+1$
$\gamma_{n+1}^0(N) = \gamma_n^0(N) - \varepsilon_n^{b\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)$	1
$\sigma_{n+1}(N) = \sigma_n(N) - \alpha_n^y(N)(\sigma_n^b(N))^{-1}\alpha_n^{y\mathrm{T}}(N)$	1
$\sigma_{n+1}^f(N-1) = \sigma_n^f(N-1) - \alpha_n^{\mathrm{T}}(N-1)(\sigma_n^b(N-1))^{-1}\alpha_n(N-1)$	1
$\sigma_{n+1}^b(N) = \sigma_n^b(N-1) - \alpha_n(N-1)(\sigma_n^f(N-1))^{-1}\alpha_n^{\mathrm{T}}(N-1)$	1
$\sigma_{n+1}^b(N-1) = \sigma_{n+1}^b(N) - \varepsilon_{n+1}^b(N)\varepsilon_{n+1}^{b\mathrm{T}}(N)/\gamma_{n+1}^0(N)$	1
$\alpha_{n+1}^y(N) = \left[\, r_M^{\mathrm{T}}(N) \,\right]_{n+1} - r_{n+1}^{\mathrm{T}}(N)\theta_{n+1}^b(N)$	$n+1$
Tabelul continuă pe următoarea pagină	

Tabelul 11.1. Algoritmul "latice a erorilor de predicţie" pentru modele multivariabile de tip sumă de convoluţie.

Scanned with OKEN Scanner

Continuarea Algoritmului 11.1.1.

$\theta^b_{n+1}(N-1) = \theta^b_{n+1}(N) - K^*_{n+1}(N)\varepsilon^{bT}_{n+1}(N)/\gamma^0_{n+1}(N)$	$n+1$
$\alpha_{n+1}(N-1) = \left[r^f_M(N-1) \right]_{n+1} - \theta^{bT}_{n+1}(N-1) r^f_{n+1}(N-1)$	1
$(*)\ \varepsilon_{n+1}(N) = \varepsilon_n(N) - \alpha^y_n(N)(\sigma^b_n(N))^{-1}\varepsilon^b_n(N)$	1
$(*)\ \hat{y}_{n+1}(N) = y(N) - \varepsilon_{n+1}(N)$	1
$\theta_{n+2}(N) = \begin{bmatrix} \theta_{n+1}(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta^b_{n+1}(N) \\ I \end{bmatrix} (\sigma^b_{n+1}(N))^{-1}\alpha^{yT}_{n+1}(N)$	$n+1$
$\theta^f_{n+2}(N-1) = \begin{bmatrix} \theta^f_{n+1}(N-1) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta^b_{n+1}(N-1) \\ I \end{bmatrix} (\sigma^b_{n+1}(N-1))^{-1}\alpha_{n+1}(N-1)$	$n+1$
$\theta^b_{n+2}(N) = \begin{bmatrix} 0 \\ \theta^b_{n+1}(N-1) \end{bmatrix} + \begin{bmatrix} I \\ -\theta^f_{n+1}(N-1) \end{bmatrix} (\sigma^f_{n+1}(N-1))^{-1}\alpha^T_{n+1}(N-1)$	$n+1$
$K^*_{n+2}(N) = \begin{bmatrix} K^*_{n+1}(N) \\ 0 \end{bmatrix} + \begin{bmatrix} -\theta^b_{n+1}(N) \\ I \end{bmatrix} (\sigma^b_{n+1}(N))^{-1}\varepsilon^b_{n+1}(N)$	$n+1$
Total (în cazul scalar): $\mathcal{O}(4M^2)$ [flops]	

Tabelul 11.2. Continuarea Algoritmului 11.1.

Utilizând algoritmul în mod bloc recursiv, adică evaluând estimaţiile parametrilor după fiecare L momente de eşantionare, rezultă un număr de operaţii/pas de eşantionare

$$n_{med} = 2n_{max} + 4n^2_{max}/L. \tag{11.11}$$

Pentru modelarea AR, care este un caz particular al modelelor de tip sumă de convoluţie, conform cu (10.5),(10.6), vectorii $r_n(N)$ şi $r^f_n(N)$ coincid, astfel încât propagarea în timp are un cost aproximativ de n_{max} operaţii.

11.2. Ferestre de date pentru urmărirea parametrilor variabili în timp

În foarte multe aplicaţii de modelare, parametrii semnalului sau sistemului studiat suferă modificări lente în timp (e.g. schimbarea parametrilor modelului dinamic al unei instalaţii datorită modificărilor lente în timp a punctului nominal de funcţionare). Algoritmii de modelare recursivi în timp, sau adaptivi, utilizează mărimile care se propagă în timp pentru memorarea "istoriei" fenomenului modelat, la fiecare moment de timp informaţia conţinută în noile date fiind extrasă şi adăugată memoriei algoritmului, parametrii modelului fiind calculaţi astfel încât să fie consistenţi atât cu istoria memorată, cât şi cu noile date. În cazul în care parametrii cu care sistemul a generat datele "vechi" diferă mult faţă de parametrii sistemului cu care se generează datele din istoria recentă, memorarea istoriei îndepărtate şi luarea în considerare a acesteia în estimarea parametrilor va conduce la înrăutăţirea calităţii estimaţiei parametrilor actuali.

Scanned with OKEN Scanner

În consecinţă, în cazul sistemelor cu parametri variabili în timp sunt necesare modalităţi pentru "uitarea" datelor "vechi", care nu mai prezintă o informaţie capabilă să îmbunătăţească estimaţia. Pentru a realiza fenomenul de deponderare a datelor ("uitare"), există un mare număr de tehnici, numite ferestre de date.

Pentru algoritmii de tip latice, cea mai comună modalitate de deponderare a datelor vechi este utilizarea factorului de uitare exponenţial, prezentată în Capitolul 10. Utilizarea altor tehnici, cum ar fi fereastra dreptunghiulară prezentată în [65], conduce la o complexitate mult mai mare a algoritmilor. În cazul algoritmului cu propagarea covarianţelor se vor putea introduce fără modificări substanţiale ale algoritmului efecte foarte diferite de ponderare, datorită simplităţii secţiuniii de propagare în timp.

Se poate modifica criteriul (10.12), pentru a putea introduce o ponderare arbitrară a erorilor, în modul următor

$$Q_{n,N}(\theta_n) = \sum_{t=0}^{N} w_N(t) v_n(t) v_n^{\mathrm{T}}(t) = \sum_{t=0}^{N} w_N(t)(y(t) - \theta_n^{\mathrm{T}}\varphi_n(t))(y(t) - \theta_n^{\mathrm{T}}\varphi_n(t))^{\mathrm{T}}. \quad (11.12)$$

Forma funcţiei de ponderare $w_N(t)$ va fi în principiu aleasă astfel ca pentru valori mici ale lui t funcţia să aibă valori mici, pentru a nu penaliza erorile modelului (10.1) când este de aşteptat ca aceste erori să fie mari, întrucât parametrii procesului au fost sensibil diferiţi faţă de cei actuali, iar pentru valori ale lui t apropiate de N, funcţia $w(t)$ să aibă valori mari, pentru ca prin minimizarea criteriului (11.12) să se obţină un model care să predicteze cât mai bine (cu erori cât mai mici) ieşirea procesului actual.

Soluţia θ_n care minimizează (11.12) rezultă în acelaşi mod ca (10.16) şi este dată de relaţia

$$\begin{aligned} \theta_n(N) &= (R_n^w(N))^{-1} r_n^w(N) = \\ &= (\sum_{t=0}^{N} w_N(t)\varphi_n(t)\varphi_n^{\mathrm{T}}(t))^{-1} \sum_{t=0}^{N} w_N(t)\varphi_n(t)y^{\mathrm{T}}(t), \end{aligned} \quad (11.13)$$

unde s-au folosit notaţiile:

$$R_n^w(N) = \sum_{t=0}^{N} w_N(t)\varphi_n(t)\varphi_n^{\mathrm{T}}(t), \quad (11.14)$$

$$r_n^w(N) = \sum_{t=0}^{N} w_N(t)\varphi_n(t)y^{\mathrm{T}}(t). \quad (11.15)$$

Se observă din (11.14),(11.15) că efectul de ponderare a erorilor în (11.12) se obţine prin ponderarea termenilor sumelor prin care se definesc intercovarianţele de eşantion ρ_k^{yu} şi autocovarianţele de eşantion ρ_k^u ce intervin ca elemente ale matricelor (11.14),(11.15). Ca urmare, pentru algoritmii cu propagarea covarianţelor vom înlocui mărimile recursive în timp (11.6)–(11.8) cu covarianţele ponderate:

$$\sigma_0^{fw}(N) = \sum_{t=0}^{N} w_N(t)u(t)u^{\mathrm{T}}(t), \quad (11.16)$$

$$r_n^{fw}(N) = \sum_{t=0}^{N} w_N(t)\varphi_n(t)u^{\mathrm{T}}(t) \quad (11.17)$$

Scanned with OKEN Scanner

şi $r_n^w(N)$ din (11.15).

În continuare vor fi prezentate câteva dintre cele mai simple ferestre de date şi relaţiile recursive de calcul a covarianţelor ponderate (11.15)–(11.17). Pentru simplificarea notaţiilor, vom defini relaţia prototip de ponderare a unei mărimi $x(t)$ pentru a deveni mărimea ponderată $a^w(N)$

$$a^w(N) = \sum_{t=0}^{N} w_N(t)x(t), \tag{11.18}$$

relaţiile (11.15)–(11.17) obţinându-se din (11.18) prin alegerea corespunzătoare a mărimilor $x(t)$ şi $a^w(N)$.

În practică se utilizează două tipuri de ferestre de date: alunecătoare, pentru care numărul de termeni diferiţi de zero în suma (11.18) este limitat la o valoare L (numită lungimea ferestrei), şi ferestre cu memorie crescătoare, pentru care contribuţia datelor vechi este deponderată dar nu este niciodată complet anulată.

Fereastra dreptunghiulară

$$w_N(t) = \begin{cases} 1 & N-L < t \leq N \\ 0 & \text{în rest} \end{cases} \tag{11.19}$$

este cea mai simplă fereastră alunecătoare. Relaţia recursivă de calcul a mărimilor ponderate rezultă imediat:

$$a^w(N) = \sum_{t=N-L+1}^{N} x(t) = a^w(N-1) + x(N) - x(N-L), \tag{11.20}$$

secţiunea recursivă în timp a algoritmului 11.1.1 devenind:

$$r_M(N) = r_M(N-1) + \varphi_M(N)y^{\mathrm{T}}(N) - \varphi_M(N-L)y^{\mathrm{T}}(N-L), \tag{11.21}$$

$$r_M^f(N) = r_M^f(N-1) + \varphi_M(N)u^{\mathrm{T}}(N) - \varphi_M(N-L)u^{\mathrm{T}}(N-L), \tag{11.22}$$

$$\sigma_0^f(N) = \sigma_0^f(N-1) + u(N)u^{\mathrm{T}}(N) - u(N-L)u^{\mathrm{T}}(N-L). \tag{11.23}$$

Din (11.21)–(11.23) rezultă modul simplu în care acţionează fereastra dreptunghiulară: la fiecare moment N de timp se ia în considerare contribuţia noilor date, această contribuţie fiind menţinută constantă de-a lungul a L paşi, iar după L paşi este anulată în întregime.

Fereastra triunghiulară se defineşte

$$w_N(t) = \begin{cases} (t+L-N)/L & N-L < t \leq N \\ 0 & \text{în rest.} \end{cases} \tag{11.24}$$

Contribuţia datelor de la pasul curent N este luată în considerare integral doar la pasul N, la fiecare nou pas de eşantionare fiind deponderată în mod liniar, astfel ca după L paşi să fie anulată în întregime. Notând suma valorilor $x(t)$, ponderate prin fereastra triunghiulară

$$b^w(N) = \sum_{t=N-L+1}^{N} \frac{1}{L}(t+L-N)x(t),$$

Scanned with OKEN Scanner

putem evidenţia următoarele relaţii recursive

$$\begin{aligned} b^w(N) &= \frac{1}{L}x(N-L+1)+\frac{2}{L}x(N-L+2)+\ldots+\frac{L-1}{L}x(N-1)+x(N)= \\ &= b^w(N-1)-\frac{1}{L}(x(N-L)+x(N-L-1)+\ldots+x(N-1))+x(N)= \\ &= b^w(N-1)+x(N)-\frac{1}{L}a^w(N-1), \end{aligned} \tag{11.25}$$

în care $a^w(N)$ satisface relaţia recursivă (11.20). Înglobarea ferestrei triunghiulare în secţiunea recursivă a Algoritmului 11.1.1 se va realiza înlocuind recursia (11.6) a lui σ_0^f prin:

$$\sigma_0^{fd}(N) = \sigma_0^{fd}(N-1)+u(N)u^{\mathrm{T}}(N)-u(N-L)u^{\mathrm{T}}(N-L), \tag{11.26}$$

$$\sigma_0^{fw} = \sigma_0^{fw}(N-1)+u(N)u^{\mathrm{T}}(N)-\frac{1}{L}\sigma_0^{fd}(N-1) \tag{11.27}$$

şi înlocuind prin grupuri asemănătoare de relaţii şi recursiile în timp pentru r_n^f şi r_n.

Fereastra Bartlett se defineşte astfel

$$w_n(t)=\begin{cases} \frac{2}{L}(t+L-N) & N-L<t\le N-\frac{L}{2} \\ \frac{2}{L}(N-t) & N-\frac{L}{2}<t<N \\ 0 & \text{în rest.} \end{cases} \tag{11.28}$$

Suma ponderată a valorilor $x(t)$ va fi notată

$$c^w(N)=\sum_{t=0}^{N}w_N(t)x(t)=\sum_{t=0}^{\infty}h(N-t)x(t), \tag{11.29}$$

mărimea $h(\tau)$ având semnificaţia unei funcţii pondere şi fiind definită prin

$$h(\tau)=\begin{cases} \frac{2}{L}\tau & 0\le\tau<\frac{L}{2} \\ \frac{2}{L}(L-\tau) & \frac{L}{2}\le\tau<L \\ 0 & \text{în rest.} \end{cases} \tag{11.30}$$

Din (11.29) rezultă că $c^w(N)$ poate fi considerat ca fiind ieşirea filtrului discret cu funcţia pondere $h(\tau)$ şi având ca intrare secvenţa $x(N)$. Funcţia de transfer a filtrului este

$$\begin{aligned} H(z) &= \mathcal{Z}[h(\tau)]=\mathcal{Z}\frac{2}{L}(\tau-2e^{-\tau L/2}\tau+e^{-\tau L}\tau)= \\ &= \frac{2}{L}\frac{z^{-1}}{(1-z^{-1})^2}(1-2z^{-L/2}+z^{-L}), \end{aligned} \tag{11.31}$$

putându-se scrie deci relaţia între intrarea şi ieşirea filtrului

$$(1-2q^{-1}+q^{-2})c^w(N)=\frac{2}{L}(q^{-1}-2q^{-L/2-1}+q^{-L-1})x(N),$$

sau

$$c^w(N)=2c^w(N-1)-c^w(N-2)+\frac{2}{L}(x(N-1)-2x(N-\frac{L}{2}-1)+x(N-L-1)). \tag{11.32}$$

Scanned with OKEN Scanner

Pentru a îngloba în Algoritmul 11.1.1 fereastra de date Bartlett, cele trei matrice de covarianţă din secţiunea recursivă în timp vor fi adaptate conform relaţiei (11.32).

Fereastra Hanning se defineşte prin ponderile

$$w_N(t) = \begin{cases} 0,5 - 0,5\cos\frac{2\pi(N-t)}{L-1} & N-L+1 \le t \le N \\ 0 & \text{în rest.} \end{cases} \tag{11.33}$$

Notând cu $a^w(N)$ suma valorilor $x(t)$ ponderate conform ferestrei Hanning, evaluarea recursivă în timp rezultă prin următorul set de relaţii[88]:

$$a^w(N) = 0,5b^w(N) - 0,5c^w(N), \tag{11.34}$$

$$b^w(N) = b^w(N-1) + x(N) - x(N-L), \tag{11.35}$$

$$c^w(N) = \cos(\frac{2\pi}{L-1})(2c^w(N-1) - x(N-1) - x(N-L)) - c^w(N-2) + x(N) - x(N-L-1). \tag{11.36}$$

Dintre ferestrele cu memorie crescătoare, cea mai cunoscută este cea exponenţială

$$w_N(t) = \begin{cases} \lambda^{N-t} & t \le N \\ 0 & \text{în rest} \end{cases} \tag{11.37}$$

cu $|\lambda| \le 1$, care a fost deja utilizată în Capitolul 10.

De notat că această fereastră constituie de fapt o filtrare trece jos a secvenţei $x(N)$, funcţia de transfer a filtrului fiind

$$H(z) = \mathcal{Z}h(\tau) = \mathcal{Z}w_n(N-\tau) = \frac{1}{1-\lambda z^{-1}}. \tag{11.38}$$

Din (11.31),(11.38) rezultă că ferestrele cu memorie crescătoare se realizează prin filtre IIR (cu răspuns infinit la impuls), pe când ferestrele alunecătoare se obţin utilizând filtre FIR (cu răspuns finit la impuls).

Fereastra cu memorie crescătoare corespunzând filtrului cu funcţia de transfer

$$H(z) = \frac{1}{(1-\lambda z^{-1})^2} \tag{11.39}$$

este numită fereastră Barnwell şi are ponderile

$$w_N(t) = \begin{cases} (N-t+1)\lambda^{N-t} & t \le N \\ 0 & \text{în rest.} \end{cases} \tag{11.40}$$

Din funcţia de transfer (11.39) rezultă imediat modul de calcul recursiv al sumei ponderate prin fereastra Barnwell

$$a^w(N) = 2\lambda a^w(N-1) - \lambda^2 a^w(N-2) + x(N). \tag{11.41}$$

Scanned with OKEN Scanner

Partea a treia

Metode Rapide Recursive în Timp

Această parte a lucrării prezintă algoritmi rapizi recursivi în timp pentru estimarea parametrilor următoarelor tipuri de modele: modele "autoregresive" (AR) şi modele multivariabile "sumă de convoluţie".
Algoritmii rapizi prezentaţi sunt de tip on-line, determinând la fiecare moment de eşantionare parametrii unui model de ordin fixat, N. Eficienţa algoritmilor atinge $\mathcal{O}(N)$ operaţii aritmetice la fiecare nouă adaptare a parametrilor, spre deosebire de algoritmii standard recursivi în timp, care necesită $\mathcal{O}(N^2)$ operaţii.

Scanned with OKEN Scanner

Capitolul 12

Algoritmi Recursivi în Timp pentru Modele AR Monovariabile

12.1. Metoda convenţională recursivă în timp a celor mai mici pătrate

În acest capitol va fi considerată problema modelării seriei $\{y(t)\}_{t=0}^{N}$ ca un proces autoregresiv de ordin cunoscut (fixat), n:

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = e(t). \tag{12.1}$$

Se notează parametrii modelului autoregresiv

$$\theta = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}^{\mathrm{T}} \tag{12.2}$$

şi vectorul observaţiilor

$$\varphi(t) = \begin{bmatrix} -y(t-1) & \ldots & -y(t-n) \end{bmatrix}^{\mathrm{T}}. \tag{12.3}$$

În Secţiunea 7.2 a fost dedusă estimaţia off-line a celor mai mici pătrate a vectorului parametrilor θ, utilizând datele $y(0), \ldots, y(N)$, criteriul de performanţă fiind covarianţa de eşantion a erorilor modelului. Această estimaţie rezultă rezolvând sistemul

$$R(N)\theta(N) = r(N), \tag{12.4}$$

în care:

$$R(N) = \sum_{t=n}^{N} \varphi(t)\varphi^{\mathrm{T}}(t), \tag{12.5}$$

$$r(N) = \sum_{t=n}^{N} \varphi(t)y(t). \tag{12.6}$$

În algoritmii recursivi în timp, estimaţia vectorului parametrilor θ poate fi cerută la toate momentele de timp $N = 1, 2, \ldots, N_{max}$ şi există astfel posibilitatea organizării eficiente a calculelor, exploatând relaţia recursivă în timp existentă între $\theta(N)$ şi $\theta(N-1)$. Conform cu (12.4), $\theta(N-1)$ este soluţia sistemului

$$R(N-1)\theta(N-1) = r(N-1), \tag{12.7}$$

Scanned with OKEN Scanner

care se rescrie, utilizând (12.5) şi (12.6),

$$\begin{aligned}(R(N)-\varphi(N)\varphi^{\mathrm{T}}(N))\theta(N-1) &= r(N)-\varphi(N)y(N)= \\ &= R(N)\theta(N)-\varphi(N)y(N),\end{aligned} \tag{12.8}$$

sau

$$R(N)\theta(N)=R(N)\theta(N-1)+\varphi(N)(y(N)-\varphi^{\mathrm{T}}(N)\theta(N-1)). \tag{12.9}$$

Definim vectorul Kalman ("aposteriori") ca soluţie a sistemului

$$R(N)K^{*}(N)=\varphi(N), \tag{12.10}$$

relaţia (12.9) scriindu-se

$$R(N)\theta(N)=R(N)(\theta(N-1)+K^{*}(N)(y(N)-\varphi^{\mathrm{T}}(N)\theta(N-1))). \tag{12.11}$$

Rezultă că o soluţie a sistemului (12.4) (sau unica soluţie dacă $R(N)$ este nesingulară) se obţine astfel

$$\theta(N)=\theta(N-1)+K^{*}(N)(y(N)-\varphi^{\mathrm{T}}(N)\theta(N-1)), \tag{12.12}$$

deci pe baza unei soluţii a sistemului (12.7), corectată printr-un termen proporţional cu eroarea de predicţie a valorii $y(N)$,

$$e(N)=y(N)-\theta^{\mathrm{T}}(N-1)\varphi(N). \tag{12.13}$$

Presupunând că numărul de date disponibile este suficient pentru ca matricea $R(N)$ să fie nesingulară (conform Sect. 7.3, $N>n+p$, unde p este primul moment pentru care $y(t)\neq 0$), vectorul Kalman va fi dat de

$$K^{*}(N)=R^{-1}(N)\varphi(N), \tag{12.14}$$

rezultând relaţia de calcul recursiv a predictorului $\theta(N)$

$$\theta(N)=\theta(N-1)+R^{-1}(N)\varphi(N)(y(N)-\varphi^{\mathrm{T}}(N)\theta(N-1)). \tag{12.15}$$

Organizarea calculelor pentru evaluarea eficientă numeric a relaţiei (12.15) este problema centrală pentru obţinerea de algoritmi recursivi în timp.

Aparent, numărul de operaţii din (12.15) este mai mare chiar decât numărul de operaţii cerut de rezolvarea off-line a sistemului (12.4) (inversarea matricei $R(N)$ cerând $\mathcal{O}(n^3)$), dar evaluând recursiv $R^{-1}(N)$ pe baza matricei $R^{-1}(N-1)$, numărul de operaţii se va reduce la $\mathcal{O}(n^2)$. Din relaţia de definiţie (12.5) rezultă

$$R(N)=R(N-1)+\varphi(N)\varphi^{\mathrm{T}}(N) \tag{12.16}$$

şi înmulţind această relaţie la stânga cu $R^{-1}(N)$ şi la dreapta cu $R^{-1}(N-1)$ se obţine

$$R^{-1}(N-1)=R^{-1}(N)+R^{-1}(N)\varphi(N)\varphi^{\mathrm{T}}(N)R^{-1}(N-1). \tag{12.17}$$

Definind vectorul Kalman "apriori"

$$K(N)=R^{-1}(N-1)\varphi(N), \tag{12.18}$$

Scanned with OKEN Scanner

Algoritmul 12.1.1.

ALGORITM CONVENŢIONAL RECURSIV ÎN TIMP		*Număr operaţii/ ecuaţie* ⋆	/	+
1. *Iniţializarea recursiilor în timp* $(N = 0)$				
$P(N_0) = \mu I \quad (\mu \gg 0);$				
$\theta(N_0) = 0$				
2. *Recursii în timp*				
Pentru $N = N_0 + 1, \ldots, N_{max}$				
$K(N) = P(N-1)\varphi(N)$	(1)	n_a^2	0	$n_a^2 - n_a$
$K^*(N) = K(N)/(1 + K^{\mathrm{T}}(N)\varphi(N))$	(2)	n_a	n_a	n_a
$P(N) = P(N-1) - K^*(N)K^{\mathrm{T}}(N)$	(3)	$\frac{n_a^2+n_a}{2}$	0	$\frac{n_a^2+n_a}{2}$
$e(N) = y(N) - \varphi^{\mathrm{T}}(N)\theta(N-1)$	(4)	n_a	0	n_a
$\theta(N) = \theta(N-1) + K^*(N)e(N)$	(5)	n_a	0	n_a
Total operaţii: $\frac{3n_a^2+7n_a}{2}$ *înmulţiri;* n_a *împărţiri;* $\frac{3n_a^2+5n_a}{2}$ *adunări*				

Tabelul 12.1. Algoritm convenţional recursiv în timp.

rezultă

$$R^{-1}(N) = R^{-1}(N-1) - K^*(N)K^{\mathrm{T}}(N). \tag{12.19}$$

Înmulţind (12.19) la dreapta cu $\varphi(N)$ se obţine

$$K^*(N) = K(N) - K^*(N)K^{\mathrm{T}}(N)\varphi(N), \tag{12.20}$$

deci între cei doi vectori Kalman există relaţia

$$K^*(N) = \frac{K(N)}{1 + K^{\mathrm{T}}(N)\varphi(N)}. \tag{12.21}$$

Notăm inversa matricei $R(N)$

$$P(N) = R^{-1}(N) \tag{12.22}$$

şi grupăm relaţiile (12.18),(12.21),(12.19), (12.13), şi (12.15) pentru a forma un algoritm recursiv în timp al estimaţiilor parametrilor, pe care îl vom numi în continuare "convenţional" şi care este prezentat în Tabelul 12.1.

Alegerea valorilor iniţiale pentru vectorul parametrilor θ şi pentru matricea $P(N)$ (proporţională cu matricea de covarianţă a estimaţiilor) este discutată în detaliu în [76], valorile considerate la pasul 1 din Algoritmul 12.1 fiind recomandate în cazul în care nu există a priori informaţie despre estimaţiile parametrilor.

Scanned with OKEN Scanner

Algoritmul convenţional recursiv în timp constituie un prim pas spre eficientizarea evaluării relaţiei de adaptare a parametrilor (12.15), obţinându-se o reducere a numărului de operaţii de la $\mathcal{O}(n^3)$ la $\mathcal{O}(n^2)$. Factorul care împiedică reducerea sub $(\mathcal{O}(n^2))$ a numărului de operaţii este utilizarea matricei $P(N-1)$ pentru obţinerea vectorului de amplificare Kalman, aşa cum rezultă din (1) şi (3) din Algoritmul 12.1. Algoritmii rapizi prezentaţi în continuare se bazează pe utilizarea unor relaţii recursive în care să nu fie implicaţi decât vectori (nu şi matrice) pentru propagarea în timp a vectorilor Kalman, rezultând astfel costuri de ordinul a n operaţii.

12.2. Algoritmul Fast Kalman

Proprietăţile de recurenţă în timp şi ordin ale estimaţiei (12.4) au fost analizate sistematic în Secţiunea 7.3, relaţiile rezultate fiind cuprinse în Tabelele 7.4 şi 7.6. Pentru deducerea algoritmilor rapizi de modelare recursivi în timp prezentaţi în acest capitol se va face referire frecventă la relaţiile din aceste tabele. În Tabelul 12.2 sunt reluate, pentru simplitatea referirii, definiţiile mărimilor introduse în Capitolul 7 care intervin în relaţiile recursive în ordin şi timp (indicele inferior desemnează dimensiunea predictorului la care se referă definiţia).

Problema fundamentală pentru obţinerea unui algoritm eficient recursiv în timp este propagarea în timp a vectorilor Kalman. După cum se poate observa în Tabelul 7.4, nu există o relaţie directă de recurenţă în timp pentru $K_n^*(N)$ sau $K_n(N)$. În algoritmul Fast Kalman[43] se utilizează numai vectorul $K_n^*(N)$, iar modul de propagare în timp al acestuia se bazează pe recursiile în ordin "impure" din Tabelul 7.6 (acele recursii în ordin în care are loc şi o modificare a argumentului de timp, fapt care a provocat probleme la algoritmii latice, dar care se dovedeşte util pentru deducerea unor algoritmi recursivi în timp) astfel:

$$K_n^*(N-1) \overset{(7.133)}{\longrightarrow} K_{n+1}^*(N) \overset{(7.131)}{\longrightarrow} K_n^*(N). \tag{12.41}$$

Pentru a nu utiliza decât un singur vector Kalman, $K_n^*(N)$, vom rescrie relaţia (7.94), ţinând cont de (7.99) şi (7.100),

$$\theta_n(N) = \theta_n(N-1) + K_n^*(N)e_n^f(N). \tag{12.42}$$

Eroarea de predicţie $e_n^f(N)$ poate fi calculată direct din definiţia (12.34). Utilizarea relaţiei (7.133)

$$K_{n+1}^*(N) = \begin{bmatrix} 0 \\ K_n^*(N-1) \end{bmatrix} - \frac{\varepsilon_n^f(N-1)}{\sigma_n(N-1)} \begin{bmatrix} 1 \\ \theta_n(N-1) \end{bmatrix} \tag{12.43}$$

implică şi calculul erorii de predicţie aposteriori ε_n^f (direct din definiţia (12.36)) şi al dispersiei erorilor de predicţie înainte, σ_n (prin relaţia recursivă (7.96)).

Trecerea de la $K_{n+1}^*(N)$ la $K_n^*(N)$ este posibilă prin utilizarea relaţiei (7.131), în care valoarea predictorului $\theta_n^b(N)$ va fi înlocuită prin (7.95) (transformat făcând uz de (7.99)(7.101))

$$\theta_n^b(N) = \theta_n^b(N-1) + K_n^*(N)e_n^b(N), \tag{12.44}$$

rezultând

$$\begin{aligned} K_{n+1}^*(N) &= \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} = \\ &= \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} \left(1 - \frac{e_n^b(N)\varepsilon_n^b(N)}{\sigma_n^b(N)}\right) - \frac{\varepsilon_n^b(N)}{\sigma_n^b(N)} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix}. \end{aligned} \tag{12.45}$$

Scanned with OKEN Scanner

Mărimi de intercorelaţie

$$R_n(N) = \sum_{t=0}^{N} \varphi_n(t)\varphi_n^{\mathrm{T}}(t) \tag{12.23}$$

$$r_n(N) = \sum_{t=0}^{N} \varphi_n(t)y(t) \tag{12.24}$$

$$r_n^b(N) = \sum_{t=0}^{N} \varphi_n(t)y(t-n-1) \tag{12.25}$$

$$\rho_m(N) = \sum_{t=0}^{N} y(t)y(t-m) \tag{12.26}$$

Predictori

$$\theta_n(N) = R_n^{-1}(N)r_n(N) \tag{12.27}$$
$$\theta_n^b(N) = R_n^{-1}(N)r_n^b(N) \tag{12.28}$$
$$K_n^*(N) = R_n^{-1}(N)\varphi_n(N) \tag{12.29}$$
$$K_n(N) = R_n^{-1}(N-1)\varphi_n(N) \tag{12.30}$$

Dispersii şi intercorelaţii ale erorilor de predicţie

$$\sigma_n(N) = \rho_0(N) - r_n^{\mathrm{T}}(N)\theta_n(N) \tag{12.31}$$
$$\sigma_n^b(N) = \rho_0(N-n-1) - r_n^{b\mathrm{T}}(N)\theta_n^b(N) \tag{12.32}$$
$$\alpha_n(N) = \rho_{n+1}(N) - r_n^{\mathrm{T}}(N)\theta_n^b(N) \tag{12.33}$$

Erori de predicţie

$$e_n^f(N) = y(N) - \varphi_n^{\mathrm{T}}(N)\theta_n(N-1) \tag{12.34}$$
$$e_n^b(N) = y(N-n-1) - \varphi_n^{\mathrm{T}}(N)\theta_n^b(N-1) \tag{12.35}$$
$$\varepsilon_n^f(N) = y(N) - \varphi_n^{\mathrm{T}}(N)\theta_n(N) \tag{12.36}$$
$$\varepsilon_n^b(N) = y(N-n-1) - \varphi_n^{\mathrm{T}}(N)\theta_n^b(N) \tag{12.37}$$

$$\gamma_n^0(N) = 1 - \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N)\varphi_n(N) \tag{12.38}$$
$$\gamma_n^*(N) = \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N)\varphi_n(N) \tag{12.39}$$
$$\gamma_n(N) = \varphi_n^{\mathrm{T}}(N)R_n^{-1}(N-1)\varphi_n(N) \tag{12.40}$$

Tabelul 12.2. Definiţii ale variabilelor utilizate în algoritmii rapizi pentru modele AR monovariabile.

Scanned with OKEN Scanner

Observând că ultimul element al acestui vector Kalman este

$$\left[K_{n+1}^*(N) \right]_{n+1} = -\frac{\varepsilon_n^b(N)}{\sigma_n^b(N)}, \tag{12.46}$$

relaţia (12.45) se poate rescrie

$$\begin{aligned} K_{n+1}^*(N) &= \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} (1 + e_n^b(N) \left[K_{n+1}^*(N) \right]_{n+1}) + \\ &+ \left[K_{n+1}^*(N) \right]_{n+1} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix}. \end{aligned} \tag{12.47}$$

Va rezulta deci relaţia de calcul a vectorului Kalman $K_n^*(N)$ plecând de la $K_{n+1}^*(N)$

$$\begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} = \left\{ K_{n+1}^*(N) - \left[K_{n+1}^*(N) \right]_{n+1} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix} \right\} \frac{1}{1 + e_n^b(N) \left[K_{n+1}^*(N) \right]_{n+1}}. \tag{12.48}$$

Eroarea de predicţie apriori $e_n^b(N)$ se poate calcula direct din definiţia (12.35). Sumarizând, relaţiile (12.43),(12.35),(12.48),(12.44),(12.34), (12.42),(12.36) şi (7.96) formează algoritmul Fast Kalman prezentat în Tabelul 12.3.

Iniţializarea algoritmului poate fi realizată în două moduri. O primă modalitate, numită iniţializare exactă, constă în pornirea algoritmului la un moment N_0 în care sunt disponibile suficiente valori ale seriei $y(t)$, astfel încât matricea $R_n(N_0)$ să fie nesingulară, şi toate mărimile care sunt propagate în timp în algoritm $(\varepsilon_n^f, \sigma_n, \theta_n, \theta_n^b, K_n^*)$ să fie unic definite. Calculul direct al acestor mărimi la momentul N_0, utilizând relaţiile de definiţie, implică un mare volum de calcule, $(\mathcal{O}(n^3))$. O altă variantă a iniţializării exacte, mult mai eficientă, este iterarea unui algoritm latice, de la momentul $N = 0$ la N_0, cu ordinul maxim n. După cum s-a demonstrat în Secţiunea 7.3, acest algoritm este valid şi atunci când matricea R_n este singulară, iar la momentul N_0 (când R_n nu mai este singulară) furnizează soluţiile exacte pentru toate variabilele propagate în secţiunea latice şi pentru predictori.

O a doua modalitate, numită iniţializare aproximativă, nu mai necesită o procedură specială de pornire a algoritmului. Valorile iniţiale date variabilelor care sunt propagate în timp sunt alese arbitrar, introducând de fapt perturbaţii ale datelor iniţiale. Cea mai frecvent utilizată iniţializare aproximativă este cea listată la Pasul 1 al algoritmului 12.2, care ar corespunde de fapt valorilor iniţiale nule ale seriei de timp, iar pentru dispersia σ_n se alege o valoare pozitivă mică (valoarea zero ar face imposibilă iterarea ecuaţiei (1) a algoritmului). Deşi această metodă este foarte simplă, utilizarea ei nu este recomandată datorită degradării performanţelor numerice ale algoritmului abservată pentru anumite valori ale parametrului μ[44].

12.3. Algoritmul FAEST

În algoritmul FAEST, foarte asemănător principial algoritmului Fast Kalman, se realizează calculul recursiv al predictorului $\theta_n(N)$ prin relaţia (7.94) din Tabelul 7.4

$$\theta_n(N) = \theta_n(N-1) + K_n(N)\varepsilon_n^f(N) \tag{12.49}$$

Scanned with OKEN Scanner

Algoritmul 12.2.1.

ALGORITMUL FAST KALMAN PENTRU MODELE AR MONOVARIABILE		*Număr operaţii/ ecuaţie*		
		$\star$	/	+
1. *Iniţializarea recursiilor în timp* $(N = 0)$				
$\varepsilon^f_{n_a}(N_0) = 0$				
$\theta_{n_a}(N_0) = \theta^b_{n_a}(N_0) = K^*_{n_a}(N_0) = 0$				
$\sigma^f_{n_a}(N_0) = \mu;\ \ \mu > 0$ *valoare arbitrară*				
2. *Recursii în timp*				
Pentru $N = N_0 + 1, \ldots, N_{max}$				
$K^*_{n_a+1}(N) = \begin{bmatrix} 0 \\ K^*_{n_a}(N-1) \end{bmatrix} - \frac{\varepsilon^f_{n_a}(N-1)}{\sigma_{n_a}(N-1)} \begin{bmatrix} 1 \\ \theta_{n_a}(N-1) \end{bmatrix}$	(1)	n_a	1	n_a
$e^b_{n_a}(N) = y(N - n_a - 1) - \varphi^{\mathrm{T}}_{n_a}(N)\theta^b_{n_a}(N-1)$	(2)	n_a	0	n_a
$\begin{bmatrix} K^*_{n_a}(N) \\ 0 \end{bmatrix} = \frac{1}{1+\varepsilon^b_{n_a}(N)\left[K^*_{n_a+1}(N)\right]_{n_a+1}} \{K^*_{n_a+1}(N) - $ $- \left[K^*_{n_a+1}(N)\right]_{n_a+1} \begin{bmatrix} \theta^b_{n_a}(N-1) \\ 1 \end{bmatrix} \}$	(3)	$2n_a + 1$	1	$n_a + 1$
$\theta^b_{n_a}(N) = \theta^b_{n_a}(N-1) + K^*_{n_a}(N)e^b_{n_a}(N)$	(4)	n_a	0	n_a
$e^f_{n_a}(N) = y(N) - \varphi^{\mathrm{T}}_{n_a}(N)\theta_{n_a}(N-1)$	(5)	n_a	0	n_a
$\theta_{n_a}(N) = \theta_{n_a}(N-1) + K^*_{n_a}(N)e^f_{n_a}(N)$	(6)	n_a	0	n_a
$\varepsilon^f_{n_a}(N) = y(N) - \varphi^{\mathrm{T}}_{n_a}(N)\theta_{n_a}(N)$	(7)	n_a	0	n_a
$\sigma_{n_a}(N) = \sigma_{n_a}(N-1) + e^f_{n_a}(N)\varepsilon^f_{n_a}(N)$	(8)	1	0	1
Total operaţii: $8n_a + 2$ *înmulţiri;* 2 *împărţiri;* $7n_a + 2$ *adunări*				

Tabelul 12.3. Algoritmul Fast Kalman pentru modele AR monovariabile.

Scanned with OKEN Scanner

şi se propagă în timp mărimile auxiliare necesare pentru evaluarea rapidă a vectorului amplificărilor Kalman $K_n(N)$, obţinându-se reducerea numărului de operaţii necesar la $5n$ înmulţiri faţă de $8n$ înmulţiri în algoritmul Fast Kalman, care propagă vectorul $K_n^*(N)$.

Numele algoritmului (Fast aposteriori Error Sequential Technique)[4] este datorat utilizării erorilor aposteriori pentru adaptarea predictorilor θ_n şi θ_n^b. Reducerea numărului de calcule se datorează în primul rând calculului diferit al erorilor de predicţie, care în algoritmul Fast Kalman necesita câte n flopi (ecuaţiile (2) şi (7) din Tabelul 12.3), iar în FAEST necesită numai câte un flop.

Pentru deducerea acestui algoritm vor fi utilizate relaţiile recursive în ordin şi timp din Tabelele 7.4 şi 7.6. Schema de propagare în timp a vectorului Kalman $K_n(N)$ este

$$K_n(N-1) \overset{(7.134)}{\longrightarrow} K_{n+1}(N) \overset{(7.132)}{\longrightarrow} K_n(N). \tag{12.50}$$

Pentru a alege mărimile necesare în algoritm, vom începe cu stabilirea mărimilor care intervin în (7.134) şi vom căuta cele mai eficiente modalităţi de calcul ale acestora, continuând până se ajunge la un set închis de relaţii recursive.

În relaţia recursivă (7.134) vom nota

$$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix} = \begin{bmatrix} \left[K_{n+1}(N) \right]_{1:n} \\ \left[K_{n+1}(N) \right]_{n+1} \end{bmatrix} \tag{12.51}$$

şi observăm că este necesar calculul lui e_n^f, care va rezulta direct din definiţia (12.34) (la un cost de n flop), iar σ_n se poate calcula eficient cu (7.96), dar implică şi calculul lui ε_n (7.100) şi γ_n. Pentru propagarea lui $\gamma_n(N)$ nu există o relaţie directă de recurenţă în timp, de aceea vor fi utilizate ecuaţiile recurente în ordin "impure" în modul următor:

$$\gamma_n(N-1) \overset{(7.138)}{\longrightarrow} \gamma_{n+1}(N) \overset{(7.136)}{\longrightarrow} \gamma_n(N).$$

Relaţia (7.136) va fi rescrisă

$$\gamma_n(N) = \gamma_{n+1}(N) - e_n^b(N)(e_n^b(N)/\sigma_n^b(N-1)). \tag{12.52}$$

Din relaţia (7.132)

$$K_{n+1}(N) = \begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} - \frac{e_n^b(N)}{\sigma_n^b(N-1)} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix} = \begin{bmatrix} \left[K_{n+1}(N) \right]_{1:n} \\ \left[K_{n+1}(N) \right]_{n+1} \end{bmatrix} \tag{12.53}$$

rezultă imediat

$$e_n^b(N) = -\sigma_n^b(N-1) \left[K_{n+1}(N) \right]_{n+1} \tag{12.54}$$

şi

$$\begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} = K_{n+1}(N) - \left[K_{n+1}(N) \right]_{n+1} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \end{bmatrix}, \tag{12.55}$$

iar (12.52) se rescrie în funcţie de mărimile disponibile

$$\gamma_n(N) = \gamma_{n+1}(N) + e_n^b(N) \left[K_{n+1}(N) \right]_{n+1}. \tag{12.56}$$

Scanned with OKEN Scanner

După determinarea lui $\gamma_n(N)$ se pot calcula erorile de predicţie aposteriori din cele apriori, utilizând : (7.100)-(7.102)

$$\varepsilon_n^f(N) = e_n^f(N)/(1+\gamma_n(N)), \tag{12.57}$$
$$\varepsilon_n^b(N) = e_n^b(N)/(1+\gamma_n(N)), \tag{12.58}$$

iar din (7.94) şi (7.95) vor rezulta noile valori ale predictorilor $\theta_n(N)$ şi $\theta_n^b(N)$.

Ecuaţiile (12.51),(7.138),(12.54)-(12.56), (12.58),(12.34),(12.57),(7.96),(7.97), (7.94),(7.95) formează secţiunea recursivă în timp a algoritmului FAEST, prezentată în Tabelul 12.4.

Pentru iniţializarea algoritmului FAEST există aceleaşi două variante prezentate pentru algoritmul Fast Kalman, fiind de preferat iniţializare exactă, prin iterarea unui algoritm latice între momentul $N = 0$ şi $N = N_0$. Astfel, mărimile necesare pentru prima iteraţie din FAEST, la momentul $N_0 + 1$, pot fi obţinute din secţiunea latice şi din cea a predictorilor, preluând valorile mărimilor de ordin n cu argument de timp corespunzător.

12.4. Algoritmul "Filtru Transversal Rapid"

În algoritmul "Filtru Transversal Rapid" (FTR) ("Fast Transversal Filter"[6]) este utilizat vectorul Kalman $K_n(N)$, ca şi în algoritmul FAEST, dar în locul mărimii $\gamma_n(N)$ se foloseşte raportul aposteriori/apriori, $\gamma_n^0(N)$, cele două mărimi fiind legate prin relaţia (7.102)

$$\gamma_n^0(N) = 1/(1+\gamma_n(N)). \tag{12.59}$$

În consecinţă, câteva dintre ecuaţiile Algoritmului 12.3.1 vor putea fi modificate astfel:

-ecuaţia (2) va fi înlocuită cu ((7.137), Tab. 7.6.)

$$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \varepsilon_n^{f2}(N-1)/\sigma_n(N-1); \tag{12.60}$$

- ecuaţia (5) va fi modificată, utilizând (12.59)

$$\gamma_n^0(N) = \left((\gamma_{n+1}^0(N))^{-1} + e_n^b(N)\left[K_{n+1}(N)\right]_{n+1}\right)^{-1} =$$
$$= \gamma_{n+1}^0(N)/\left(1 + e_n^b(N)\left[K_{n+1}(N)\right]_{n+1}\gamma_{n+1}^0(N)\right); \tag{12.61}$$

-ecuaţiile (6) şi (8) vor fi înlocuite, conform (12.59), prin:

$$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N), \tag{12.62}$$
$$\varepsilon_n^f(N) = e_n^f(N)\gamma_n^0(N). \tag{12.63}$$

Restul ecuaţiilor din Algoritmul 12.3.1. rămân neschimbate, rezultând că, practic, costul algoritmului FTR, prezentat în Tabelul 12.5., este acelaşi cu cel al algoritmului FAEST.

Definim următoarea "variabilă de supraveghere"[6]

$$\delta = 1 + e_n^b(N)\left[K_{n+1}(N)\right]_{n+1}\gamma_{n+1}^0(N), \tag{12.64}$$

Scanned with OKEN Scanner

Algoritmul 12.3.1.

ALGORITMUL FAEST PENTRU MODELE AR MONOVARIABILE		*Număr operaţii/ ecuaţie* ⋆ / +
1. *Iniţializarea recursiilor în timp* $(N = 0)$		
2. *Recursii în timp*		
Pentru $N = N_0 + 1, \ldots, N_{max}$		
$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix}$	(1)	n 1 n
$\gamma_{n+1}(N) = \gamma_n(N-1) + e_n^{f2}(N-1)/\sigma_n(N-2)$	(2)	1 1 1
$e_n^b(N) = -\left[K_{n+1}(N) \right]_{n+1} \sigma_n^b(N-1)$	(3)	1 0 0
$K_n(N) = \left[K_{n+1}(N) \right]_{1:n} - \left[K_{n+1}(N) \right]_{n+1} \theta_n^b(N-1)$	(4)	n 0 n
$\gamma_n(N) = \gamma_{n+1}(N) + e_n^b(N) \left[K_{n+1}(N) \right]_{n+1}$	(5)	1 0 1
$\varepsilon_n^b(N) = e_n^b(N)/(1+\gamma_n(N))$	(6)	0 1 1
$e_n^f(N) = y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N)$	(7)	n 0 n
$\varepsilon_n^f(N) = e_n^f(N)/(1+\gamma_n(N))$	(8)	0 1 1
$\sigma_n(N) = \sigma_n(N-1) + e_n^f(N)\varepsilon_n^f(N)$	(9)	1 0 1
$\sigma_n^b(N) = \sigma_n^b(N-1) + e_n^b(N)\varepsilon_n^b(N)$	(10)	1 0 1
$\theta_n^b(N) = \theta_n^b(N-1) + K_n(N)\varepsilon_n^b(N)$	(11)	n 0 n
$\theta_n(N) = \theta_n(N-1) + K_n(N)\varepsilon_n^f(N)$	(12)	n 0 n
Total operaţii: $5n+5$ *înmulţiri;* 4 *împărţiri;* $5n+6$ *adunări*		

Tabelul 12.4. Algoritmul FAEST pentru modele AR monovariabile.

Scanned with OKEN Scanner

a cărui evaluare se realizează în ecuaţia (12.61) din algoritmul FTR. Din (12.61),(7.135) şi (7.97) rezultă

$$\delta = \frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N)} = 1 - \frac{\varepsilon_n^{b2}(N)}{\sigma_n^b(N)\gamma_n^0(N)} = 1 - \frac{e_n^b(N)\varepsilon_n^b(N)}{\sigma_n^b(N)} = \\ = \frac{\sigma_n^b(N-1)}{\sigma_n^b(N)} \tag{12.65}$$

şi conform definiţiei dispersiei de eşantion σ_n^b, va rezulta pozitivitatea mărimii δ.

Datorită faptului că în algoritmul FTR evaluarea mărimii δ nu se realizează prin (12.65) (care ar garanta pozitivitatea lui δ) ci prin (12.64) în care intervin trei mărimi, eventuala deteriorare datorită erorilor numerice a acestor trei mărimi poate cunduce şi la valori negative pentru δ. S-a observat [6] că imediat după ce δ devine negativ apare fenomenul de divergenţă a estimaţiilor parametrilor, astfel încât pentru a reveni la funcţionarea normală, algoritmul trebuie reiniţializat. Deoarece δ anticipează deteriorarea performanţelor, estimaţiile parametrilor în momentul în care δ a devenit negativ constituie valori iniţiale utile la repornirea algoritmului. Procedurile de repornire făcând uz de această informaţie sunt destul de laborioase faţă de iniţializarea obişnuită (din condiţii iniţiale zero), iar performanţele obţinute prin cele două metode sunt destul de apropiate[6].

12.5. Algoritmul "Filtru Transversal Rapid" normalizat

În Capitolul 8 a fost ilustrată normalizarea algoritmilor latice, operaţie care conduce la îmbunătăţirea performanţelor numerice ale algoritmilor. Extinderea operaţiei de normalizare la algoritmii rapizi recursivi în timp[5],[6] va fi prezentată în această secţiune. În continuare vor fi utilizate definiţiile mărimilor normalizate introduse în Capitolul 8; pentru mai simpla lor referire, definiţiile au fost grupate în Tabelul 12.6. împreună cu câteva relaţii de bază care leagă erorile normalizate de mărimile γ_n^0, σ_n şi σ_n^b, deduse în Capitolul 8 (relaţiile (8.23),(8.24),(8.30) şi (8.33)).

Pentru transformarea ecuaţiilor algoritmului FTR nenormalizat în varianta normalizată, vor fi utilizate acolo unde este posibil relaţiile deduse în Capitolul 8 pentru normalizarea algoritmului latice.

Prima ecuaţie din Algoritmul 12.4.1. se poate scrie

$$\frac{1}{\sqrt{\gamma_{n+1}^0(N)}}\overline{K}_{n+1}^*(N) = \begin{bmatrix} 0 \\ \overline{K}_n^*(N-1) \end{bmatrix} \frac{1}{\sqrt{\gamma_n^0(N-1)}} - \\ - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} \overline{\theta}_n(N-2) \\ 0 \end{bmatrix} \sqrt{\sigma_n(N-2)}, \tag{12.73}$$

sau

$$\overline{K}_{n+1}^*(N) = \begin{bmatrix} 0 \\ \overline{K}_n^*(N-1) \end{bmatrix} \sqrt{\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N-1)}} - \\ - \frac{e_n^f(N-1)}{\sqrt{\sigma_n(N-1)}} \sqrt{\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N-1)}} \sqrt{\gamma_n^0(N-1)} \sqrt{\frac{\sigma_n(N-1)}{\sigma_n(N-2)}} \begin{bmatrix} \overline{\theta}_n(N-2) \\ 0 \end{bmatrix} = \\ = \left(1 - \overline{\varepsilon}_n^{f2}(N-1)\right)^{\frac{1}{2}} \begin{bmatrix} 0 \\ \overline{K}_n^*(N-1) \end{bmatrix} - \overline{\varepsilon}_n^f(N-1) \begin{bmatrix} \overline{\theta}_n(N-2) \\ 0 \end{bmatrix}. \tag{12.74}$$

Scanned with OKEN Scanner

Algoritmul 12.4.1.

ALGORITMUL FTR PENTRU MODELE AR MONOVARIABILE		*Număr operaţii/ ecuaţie* ⋆ / +
1. *Iniţializarea recursiilor în timp* $(N = 0)$		
2. *Recursii în timp*		
Pentru $N = N_0 + 1, \ldots, N_{max}$		
$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} - \frac{e_n^f(N-1)}{\sigma_n(N-2)} \begin{bmatrix} 1 \\ \theta_n(N-2) \end{bmatrix}$	(1)	n 1 n
$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \varepsilon_n^{f2}(N-1)/\sigma_n(N-1)$	(2)	1 1 1
$e_n^b(N) = -\left[K_{n+1}(N) \right]_{n+1} \sigma_n^b(N-1)$	(3)	1 0 0
$K_n(N) = \left[K_{n+1}(N) \right]_{1:n} - \left[K_{n+1}(N) \right]_{n+1} \theta_n^b(N-1)$	(4)	n 0 n
$\gamma_n^0(N) = \gamma_{n+1}^0(N) / \left(1 + e_n^b(N) \left[K_{n+1}(N) \right]_{n+1} \gamma_{n+1}^0(N)\right)$	(5)	2 1 1
$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N)$	(6)	1 0 0
$e_n^f(N) = y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N)$	(7)	n 0 n
$\varepsilon_n^f(N) = e_n^f(N)\gamma_n^0(N)$	(8)	1 0 0
$\sigma_n(N) = \sigma_n(N-1) + e_n^f(N)\varepsilon_n^f(N)$	(9)	1 0 1
$\sigma_n^b(N) = \sigma_n^b(N-1) + e_n^b(N)\varepsilon_n^b(N)$	(10)	1 0 1
$\theta_n^b(N) = \theta_n^b(N-1) + K_n(N)\varepsilon_n^b(N)$	(11)	n 0 n
$\theta_n(N) = \theta_n(N-1) + K_n(N)\varepsilon_n^f(N)$	(12)	n 0 n
Total operaţii: $5n+7$ *înmulţiri;* 3 *împărţiri;* $5n+4$ *adunări*		

Tabelul 12.5. Algoritmul FTR pentru modele AR monovariabile.

Scanned with OKEN Scanner

Erori de predicţie normalizate

$$\overline{\varepsilon}_n^f(N) = \frac{\varepsilon_n^f(N)}{\sqrt{\sigma_n(N)\gamma_n^0(N)}} = \frac{e_n^f(N)}{\sqrt{\sigma_n(N)}}\sqrt{\gamma_n^0(N)} \tag{12.66}$$

$$\overline{\varepsilon}_n^b(N) = \frac{\varepsilon_n^b(N)}{\sqrt{\sigma_n^b(N)\gamma_n^0(N)}} = \frac{e_n^b(N)}{\sqrt{\sigma_n^b(N)}}\sqrt{\gamma_n^0(N)} \tag{12.67}$$

Predictori normalizaţi

$$\overline{\theta}_n(N) = \begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} \frac{1}{\sqrt{\sigma_n(N)}} \tag{12.68}$$

$$\overline{\theta}_n^b(N) = \begin{bmatrix} \theta_n^b(N) \\ 1 \end{bmatrix} \frac{1}{\sqrt{\sigma_n^b(N)}} \tag{12.69}$$

$$\overline{K}_n^*(N) = \begin{bmatrix} K_n^*(N) \\ 0 \end{bmatrix} \frac{1}{\sqrt{\gamma_n^0(N)}} = \begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} \sqrt{\gamma_n^0(N)} \tag{12.70}$$

Relaţii de bază

$$\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N)} = \frac{\sigma_n^b(N-1)}{\sigma_n^b(N)} = 1 - \overline{\varepsilon}_n^{b2}(N) \tag{12.71}$$

$$\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N-1)} = \frac{\sigma_n(N-2)}{\sigma_n(N-1)} = 1 - \overline{\varepsilon}_n^{f2}(N-1) \tag{12.72}$$

Tabelul 12.6. Definiţii ale mărimilor normalizate şi relaţii de bază (modele AR monovariabile).

Scanned with OKEN Scanner

Ecuaţia a doua din Algoritmul 12.4.1. se rescrie imediat sub formă normalizată utilizând (12.72), iar ecuaţia a treia se modifică astfel:

$$\begin{aligned} e_n^b(N) &= \sqrt{\frac{\sigma_n^b(N)}{\gamma_n^0(N)}}\bar{\varepsilon}_n^b(N) = -\left[\, K_{n+1}(N) \,\right]_{n+1} \sigma_n^b(N-1) = \\ &= -\left[\, \overline{K}_{n+1}^*(N) \,\right]_{n+1} \frac{\sigma_n^b(N-1)}{\sqrt{\gamma_{n+1}^0(N)}}, \end{aligned} \tag{12.75}$$

rezultând

$$\begin{aligned} \bar{\varepsilon}_n^b(N) &= -\left[\, \overline{K}_{n+1}^*(N) \,\right]_{n+1} \sqrt{\frac{\gamma_n^0(N)}{\gamma_{n+1}^0(N)}} \sqrt{\frac{\sigma_n^b(N-1)}{\sigma_n^b(N)}} \sqrt{\sigma_n^b(N-1)} = \\ &= -\left[\, \overline{K}_{n+1}^*(N) \,\right]_{n+1} \sqrt{\sigma_n^b(N-1)}. \end{aligned} \tag{12.76}$$

Normalizarea ecuaţiei a patra din algoritmul FTR se realizează în următorul mod:

$$\begin{aligned} \begin{bmatrix} \overline{K}_n^*(N) \\ 0 \end{bmatrix} &= \sqrt{\gamma_n^0(N)} \begin{bmatrix} K_n(N) \\ 0 \\ 0 \end{bmatrix} = \\ &= \sqrt{\gamma_n^0(N)} \left\{ \begin{bmatrix} K_{n+1}(N) \\ 0 \end{bmatrix} + \frac{e_n^b(N)}{\sigma_n^b(N-1)} \begin{bmatrix} \theta_n^b(N-1) \\ 1 \\ 0 \end{bmatrix} \right\} = \\ &= \sqrt{\gamma_n^0(N)} \left\{ \frac{1}{\sqrt{\gamma_{n+1}^0(N)}} \overline{K}_{n+1}^*(N) + \frac{e_n^b(N)}{\sigma_n^b(N-1)} \sqrt{\sigma_n^b(N-1)} \begin{bmatrix} \overline{\theta}_n^b(N-1) \\ 0 \end{bmatrix} \right\} = \\ &= \sqrt{\frac{\gamma_n^0(N)}{\gamma_{n+1}^0(N)}} \overline{K}_{n+1}^*(N) + \sqrt{\frac{\sigma_n^b(N-1)}{\sigma_n^b(N)}} \bar{\varepsilon}_n^b(N) \begin{bmatrix} \overline{\theta}_n^b(N-1) \\ 0 \end{bmatrix} = \\ &= (1 - \bar{\varepsilon}_n^{b2}(N))^{-\frac{1}{2}} \left\{ \overline{K}_{n+1}^*(N) + \bar{\varepsilon}_n^b(N) \begin{bmatrix} \overline{\theta}_n^b(N-1) \\ 0 \end{bmatrix} \right\}. \end{aligned} \tag{12.77}$$

Ecuaţia (5) din Algoritmul 12.4.1., pentru calculul mărimii $\gamma_n^0(N)$ în funcţie de $\gamma_{n+1}^0(N)$, va căpăta o formă mult mai simplă, utilizând (12.71)

$$\gamma_n^0(N) = \gamma_{n+1}^0(N)(1 - \bar{\varepsilon}_n^{b2}(N))^{-1}. \tag{12.78}$$

Calculul erorii de predicţie înainte (ecuaţia (7) în Tabelul 12.5.) poate fi realizat astfel

$$\begin{aligned} e_n^f(N) &= y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N) = \begin{bmatrix} 1 & \theta_n^{\mathrm{T}}(N-1) \end{bmatrix} \begin{bmatrix} y(N) \\ \varphi_n(N) \end{bmatrix} = \\ &= \overline{\theta}_n^{\mathrm{T}}(N-1)\varphi_{n+1}(N+1)\sqrt{\sigma_n(N)} \end{aligned} \tag{12.79}$$

şi introducând în (12.66) rezultă

$$\bar{\varepsilon}_n^f(N) = \frac{e_n^f(N)}{\sqrt{\sigma_n(N)}} \sqrt{\gamma_n^0(N)} = \overline{\theta}_n^{\mathrm{T}}(N-1)\varphi_{n+1}(N+1)\sqrt{\gamma_n^0(N)}. \tag{12.80}$$

Scanned with OKEN Scanner

Algoritmul 12.5.1.

ALGORITMUL FTR NORMALIZAT

1. *Iniţializarea recursiilor în timp* $(N = 0)$
2. *Recursii în timp*

Pentru $N = N_0 + 1, \ldots, N_{max}$

$$\overline{K}^*_{n+1}(N) = (1 - \overline{\varepsilon}^{f2}_n(N-1))^{\frac{1}{2}} \left\{ \begin{bmatrix} 0 \\ \overline{K}^*_n(N-1) \end{bmatrix} - \overline{\varepsilon}^f_n(N-1) \begin{bmatrix} \overline{\theta}_n(N-2) \\ 0 \end{bmatrix} \right\} \quad (1)$$

$$\sqrt{\gamma^0_{n+1}(N)} = \sqrt{\gamma^0_n(N-1)}(1 - \overline{\varepsilon}^{f2}_n(N-1))^{1/2} \quad (2)$$

$$\overline{\varepsilon}^b_n(N) = -\left[\overline{K}^*_{n+1}(N)\right]_{n+1} \sqrt{\sigma^b_n(N-1)} \quad (3)$$

$$\begin{bmatrix} \overline{K}^*_n(N) \\ 0 \end{bmatrix} = (1 - \overline{\varepsilon}^{b2}_n(N))^{-\frac{1}{2}} \left\{ \overline{K}^*_{n+1}(N) + \overline{\varepsilon}^b_n(N) \begin{bmatrix} \overline{\theta}^b_n(N-1) \\ 0 \end{bmatrix} \right\} \quad (4)$$

$$\sqrt{\gamma^0_n(N)} = \sqrt{\gamma^0_{n+1}(N)}(1 - \overline{\varepsilon}^{b2}_n(N))^{-\frac{1}{2}} \quad (5)$$

$$\overline{\varepsilon}^f_n(N) = \overline{\theta}^{\mathrm{T}}_n(N-1)\varphi_{n+1}(N+1)\sqrt{\gamma^0_n(N)} \quad (6)$$

$$\sqrt{\sigma^b_n(N)} = \sqrt{\sigma^b_n(N-1)}(1 - \overline{\varepsilon}^{b2}_n(N))^{-\frac{1}{2}} \quad (7)$$

$$\begin{bmatrix} \overline{\theta}_n(N) \\ 0 \end{bmatrix} = (1 - \overline{\varepsilon}^{f2}_n(N))^{\frac{1}{2}} \begin{bmatrix} \overline{\theta}_n(N-1) \\ 0 \end{bmatrix} + \overline{\varepsilon}^f_n(N) \begin{bmatrix} 0 \\ \overline{K}^*_n(N) \end{bmatrix} \quad (8)$$

$$\overline{\theta}^b_n(N) = (1 - \overline{\varepsilon}^{b2}_n(N))^{\frac{1}{2}} \overline{\theta}^b_n(N-1) + \overline{\varepsilon}^b_n(N)\overline{K}^*_n(N) \quad (9)$$

$$\begin{bmatrix} 1 \\ \theta_n(N) \end{bmatrix} = \left(\left[\overline{\theta}_n(N)\right]_1\right)^{-1} \overline{\theta}_n(N) \quad (10)$$

Tabelul 12.7. Algoritmul FTR pentru modele AR monovariabile.

Calculul dispersiei de eşantion $\sigma_n(N)$ (relaţia (9), Tabelul 12.5.) nu mai este necesar în varianta normalizată; dispersia $\sigma^b_n(N)$, cerută în (12.77), se poate evalua din (12.71)

$$\sigma^b_n(N) = \sigma^b_n(N-1)(1 - \overline{\varepsilon}^{b2}_n(N))^{-1}. \quad (12.81)$$

Pentru recursia în timp a predictorilor vor fi utilizate relaţiile (8.58) şi (8.59) deduse în Capitolul 8, iar pentru trecerea de la predictorul normalizat la cel nenormalizat se va utiliza relaţia (8.65).

În Tabelul 12.7. au fost grupate relaţiile care formează algoritmul FTR normalizat, scrise astfel încât numărul de extrageri de radical să fie minim (de exemplu, în loc să se calculeze $\gamma^0_n(N)$ cu relaţia (12.78) s-a preferat forma din ecuaţia (5), în algoritm fiind necesară numai variabila $\sqrt{\gamma^0_n(N)}$).

Scanned with OKEN Scanner

Capitolul 13

Algoritmul "Filtru Transversal Rapid" pentru Modele Multivariabile

13.1. Algoritmi recursivi în timp pentru modele multivariabile

În Capitolul 10 a fost considerată problema de modelare multivariabilă cu modele de tip sumă de convoluţie, definindu-se criteriul (matricea de covarianţă de eşantion a erorilor modelului, ponderate exponenţial) şi predictorii asociaţi, a fost prezentată soluţia care minimizează criteriul matriceal, iar din studierea proprietăţilor de recurenţă în ordin şi timp ale soluţiei au rezultat relaţiile recursive din Tabelele 10.2. şi 10.4.

În acest capitol vor fi prezentaţi algoritmi recursivi în timp pentru estimarea parametrilor modelelor multivariabile de tip sumă de convoluţie pentru un ordin n fixat, algoritmi mai eficienţi decât cei latice pentru acest tip de problemă, necesitând numai $\mathcal{O}(nq^3)$ operaţii faţă de $\mathcal{O}(n^2q^3)$ operaţii cerute de algoritmii din Secţiunea 10.2.

În Capitolul 10 s-a observat că schema de modelare rapidă pentru modelele de tip sumă de convoluţie cuprinde două etape: în prima etapă se modelează autoregresiv semnalul de intrare $u(t)$ prin predictorii înainte şi înapoi θ_n^f şi θ_n^b; în a doua etapă se construieşte estimaţia parametrilor θ_n pe baza acestor predictori autoregresivi şi a unor mărimi auxiliare. Vom urma aceleaşi etape, pentru construirea de algoritmi recursivi în timp, selectând ecuaţii din Tabelele 10.2 şi 10.4 şi completându-le apoi cu relaţiile necesare pentru obţinerea de algoritmi.

13.2. Algoritmul "Filtru Transversal Rapid"

Modelarea autoregresivă a semnalului $u(t)$ se va realiza prin extinderea la cazul multivariabil a procedurii "Filtru Transversal Rapid" prezentată pentru cazul monovariabil în Secţiunea 12.4 (de remarcat o uşoară modificare a notaţiilor– de exemplu $\theta_n \to \theta_n^f$ şi $\sigma_n \to \sigma_n^f$– faţă de Secţiunea 12.4 şi a semnului unor mărimi, provenind din convenţiile diferite făcute în Capitolul 7 pentru modelele AR şi în Capitolul 10 pentru modelele de tip sumă de convoluţie).

Unele dintre ecuaţiile din Tabelele 10.2, 10.4 furnizează, aproape fără modificări, ecuaţiile necesare secţiunii AR în ordinea sugerată de ecuaţiile de mai jos din algoritmul FTR (Tabelul 12.5) care au drept corespondent următoarele relaţii: (1) $\leftarrow$ (10.62); (2) $\leftarrow$ (10.73); (6) $\leftarrow$ (10.94); (8) $\leftarrow$ (10.93); (9) $\leftarrow$ (10.98); (10) $\leftarrow$ (10.99); (11) $\leftarrow$ (10.91); (12) $\leftarrow$ (10.90). Relaţia (7)

Scanned with OKEN Scanner

pentru calculul erorii de predicţie a intrării $u(N)$ se va scrie (conform definiţiei (10.37))

$$e_n^f(N) = u(N) - \theta_n^{f\mathrm{T}}(N-1)\varphi(N).$$

Corespondentul multivariabil al relaţiilor (3),(4) şi (5) din Algoritmul 12.4.1. va fi dedus în cele ce urmează.

Din (10.61) rezultă

$$\left[K_{n+1}(N) \right]_{n+1} = (\sigma_n^b(N-1))^{-1} e_n^b(N)/\lambda, \tag{13.1}$$

sau

$$e_n^b(N) = \lambda \sigma_n^b(N-1) \left[K_{n+1}(N) \right]_{n+1}, \tag{13.2}$$

care va corespunde ecuaţiei (3) (Tab. 12.5). Relaţiei (4) (Tab.12.5) îi va corespunde, conform (13.2) şi (10.61),

$$\begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} = K_{n+1}(N) - \begin{bmatrix} -\theta_n^b(N-1) \\ I \end{bmatrix} \left[K_{n+1}(N) \right]_{n+1}. \tag{13.3}$$

Din relaţia (10.99)

$$\sigma_n^b(N) = \lambda \sigma_n^b(N-1) + \frac{\varepsilon_n^b(N)\varepsilon_n^{b\mathrm{T}}(N)}{\gamma_n^0(N)} \tag{13.4}$$

înmulţită la dreapta cu $(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)$ şi la stânga cu $(\sigma_n^b(N-1))^{-1}$ rezultă

$$\begin{aligned} (\sigma_n^b(N-1))^{-1}\varepsilon_n^b(N) &= \lambda(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) + \\ &+ \frac{(\sigma_n^b(N-1))^{-1}\varepsilon_n^b(N)\varepsilon_n^{b\mathrm{T}}(N)(\sigma_n^b(N))^{-1}\varepsilon_n^b(N)}{\gamma_n^0(N)} = \\ &= \lambda(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) + (\sigma_n^b(N-1))^{-1}\varepsilon_n^b(N)\frac{\gamma_n^0(N) - \gamma_{n+1}^0(N)}{\gamma_n^0(N)} \end{aligned} \tag{13.5}$$

sau

$$(\sigma_n^b(N))^{-1}\varepsilon_n^b(N) = \frac{1}{\lambda}(\sigma_n^b(N-1))^{-1}\varepsilon_n^b(N)\frac{\gamma_{n+1}^0(N)}{\gamma_n^0(N)}. \tag{13.6}$$

Introducând (13.2) şi (13.6) în (10.72) rezultă

$$\gamma_{n+1}^0(N) = \gamma_n^0(N) - e_n^{b\mathrm{T}}(N) \left[K_{n+1}(N) \right]_{n+1} \gamma_{n+1}^0(N)\gamma_n^0(N), \tag{13.7}$$

de unde

$$\gamma_n^0(N) = \gamma_{n+1}^0(N)/(1 - e_n^{b\mathrm{T}}(N) \left[K_{n+1}(N) \right]_{n+1} \gamma_{n+1}^0(N)), \tag{13.8}$$

care este corespondentul căutat al relaţiei (5) (Tab. 12.5).

Secţiunea de modelare a intrării $u(t)$ dedusă mai sus serveşte de fapt calculului în $\mathcal{O}(n)$ operaţii a vectorului $K_n(N)$.

Ecuaţiile corespunzătoare calculului predictorului $\theta_n(N)$ al ieşirii $y(N)$ a modelului vor fi următoarele: relaţia de definiţie (10.36)

$$e_n(N) = y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N)$$

Scanned with OKEN Scanner

Algoritmul 13.2.1.

ALGORITMUL FTR PENTRU
MODELE MULTIVARIABILE TIP SUMĂ DE CONVOLUTIE

1. *Inițializarea recursiilor în timp* $(N = 0)$
2. *Recursii în timp*

Pentru $N = N_0 + 1, \ldots, N_{max}$

$$K_{n+1}(N) = \begin{bmatrix} 0 \\ K_n(N-1) \end{bmatrix} + \begin{bmatrix} I \\ \theta_n^f(N-2) \end{bmatrix} (\sigma_n^f(N-2))^{-1} e_n^f(N-1)/\lambda \quad (1)$$

$$\gamma_{n+1}^0(N) = \gamma_n^0(N-1) - \varepsilon_n^{f\mathrm{T}}(N-1)(\sigma_n(N-1))^{-1}\varepsilon_n^f(N-1) \quad (2)$$

$$e_n^b(N) = \lambda(\sigma_n^b(N-1)) \left[K_{n+1}(N) \right]_{n+1} \quad (3)$$

$$\begin{bmatrix} K_n(N) \\ 0 \end{bmatrix} = K_{n+1}(N) - \begin{bmatrix} -\theta_n^b(N-1) \\ I \end{bmatrix} \left[K_{n+1}(N) \right]_{n+1} \quad (4)$$

$$\gamma_n^0(N) = \frac{\gamma_{n+1}^0(N)}{1 - \gamma_{n+1}^0(N) e_n^{b\mathrm{T}}(N) \left[K_{n+1}(N) \right]_{n+1}} \quad (5)$$

$$\varepsilon_n^b(N) = e_n^b(N)\gamma_n^0(N) \quad (6)$$

$$e_n^f(N) = u(N) - \theta_n^{f\mathrm{T}}(N-1)\varphi_n(N) \quad (7)$$

$$\varepsilon_n^f(N) = e_n^f(N)\gamma_n^0(N) \quad (8)$$

$$\sigma_n^f(N) = \lambda\sigma_n^f(N-1) + e_n^f(N)\varepsilon_n^{f\mathrm{T}}(N) \quad (9)$$

$$\sigma_n^b(N) = \lambda\sigma_n^b(N-1) + e_n^b(N)\varepsilon_n^{b\mathrm{T}}(N) \quad (10)$$

$$\theta_n^f(N) = \theta_n^f(N-1) + K_n(N)\varepsilon_n^f(N) \quad (11)$$

$$\theta_n^b(N) = \theta_n^b(N-1) + K_n(N)\varepsilon_n^b(N) \quad (12)$$

$$e_n(N) = y(N) - \theta_n^{\mathrm{T}}(N-1)\varphi_n(N) \quad (13)$$

$$\varepsilon_n(N) = e_n(N)\gamma_n^0(N) \quad (14)$$

$$\sigma_n(N) = \lambda\sigma_n(N-1) + e_n(N)\varepsilon_n^{\mathrm{T}}(N) \quad (15)$$

$$\theta_n(N) = \theta_n(N-1) + K_n(N)\varepsilon_n^{\mathrm{T}}(N) \quad (16)$$

Tabelul 13.1. Algoritmul FTR pentru modele multivariabile tip sumă de convoluție.

Scanned with OKEN Scanner

şi relaţiile (10.92),(10.97) şi (10.89).

În Tabelul 13.1. este prezentat algoritmul recursiv în timp rezultat. Pentru iniţializare se recomandă iterarea algoritmului latice până matricea $R_n(N)$ devine nesingulară şi apoi iniţializarea variabilelor FTR cu valorile corespunzătoare din algoritmul latice (iniţializare exactă). După cum s-a arătat în Capitolul 10, clasa de modele de tip sumă de convoluţie poate fi uşor particularizată la alte clase de modele multivariabile (modele de autoregresie, ecuaţii cu diferenţe) astfel încât Algoritmul 13.1 poate fi uşor aplicat la o varietate mare de modele.

Scanned with OKEN Scanner

Scanned with OKEN Scanner

Anexe

Algoritmi Rapizi Implementaţi în PASCAL şi MATLAB

A.1. Proceduri PASCAL pentru implementarea algoritmilor rapizi de modelare pentru modele multivariable tip sumă de convoluţie

În această secţiune prezentăm o bibliotecă de proceduri scrise în limbajul Pascal, pentru implementarea următorilor algoritmi:

Mv_latice_FIR Algoritmul 10.1.1. (Tabelele 10.5 şi 10.6) "Latice a erorilor de predicţie şi predictorilor" pentru modele multivariabile de tip sumă de convoluţie. Iniţializarea procedurii la momentul $N = 0$ este realizată de procedura *Initialize_mv_latice*.

Mv_cov_update_FIR Algoritmul 11.1.1. (Tabelele 11.1 şi 11.2) "Algoritm bloc recursiv cu propagarea covarianţelor". Iniţializarea procedurii la momentul $N = 0$ este realizată de procedura *Initialize_mv_covar*.

```
unit FastMFIR ;
interface
const numax = 4 ;nymax = 2; nbmax = 5;
type
   u_type = array [1..numax] of real ;
   y_type = array [1..nymax] of real;
   sigmaf_type  = array [1..numax , 1..numax] of real ;
   sigma_type   = array [1..nymax , 1..nymax] of real ;
   alfay_type   = array [1..nymax , 1..numax] of real ;
   teta_type    = array [1..numax , 1..nymax] of real ;
   fi_type      = array [0..nbmax+1] of u_type;
   mv_teta_type = array[0..nbmax] of teta_type;
   mv_tetaf_type = array[0..nbmax] of sigmaf_type;
   r_M_type = array[1..nbmax] of teta_type ;
   rf_M_type = array[1..nbmax] of sigmaf_type ;
```

```
  mv_latice_type = record
     alfay : array [0..nbmax] of alfay_type ;
     alfa  : array [0..nbmax] of sigmaf_type;
     sigma : array [0..nbmax] of sigma_type ;
     sigmaf: array [0..nbmax] of sigmaf_type;
     sigmab: array [0..nbmax] of sigmaf_type;
     eps   : array [0..nbmax] of y_type ;
     epsf  : array [0..nbmax] of u_type ;
     epsb  : array [0..nbmax] of u_type ;
     gama0 : array [0..nbmax] of real ;
     ky    : mv_teta_type ;
     kf    : mv_tetaf_type;
     kb    : mv_tetaf_type;
     kf_time_kb : mv_tetaf_type ;
     Keb   : fi_type ;
    end;
  mv_direct_type = record
     teta  : array [1..nbmax] of mv_teta_type ;
     tetaf : array [1..nbmax] of mv_tetaf_type ;
     tetab : array [1..nbmax] of mv_tetaf_type ;
     kstar : array [1..nbmax] of fi_type ;
   end;

 Procedure Mv_latice_FIR ( nb,ny,nu : integer ; y_current : y_type ;
   u_current : u_type ; lambda : real ;
   var current_latice_par : mv_latice_type ;
   var current_direct_par : mv_direct_type ;
   option_direct_par: boolean );

 Procedure Initialize_mv_latice( nb,ny,nu : integer ;
var current_latice_par : mv_latice_type ) ;

 Procedure Mv_cov_update_FIR ( nb,ny,nu : integer ; y_current : y_type;
u_current : u_type ;
var r_M : r_M_type ; var rf_M : rf_M_type;
var fi : fi_type ;
var current_latice_par : mv_latice_type ;
var current_direct_par : mv_direct_type ;
option_order_recursions : boolean );

 Procedure Initialize_mv_covar( nb,ny,nu : integer ;
var r_M : r_M_type; var rf_M : rf_M_type;
var fi : fi_type ;
var current_latice_par : mv_latice_type );
```

Scanned with OKEN Scanner

```
  implementation
  uses Printer;
  function pseudo_inv_scalar (delta : real) : real;
begin
if not ((delta > -1.E-10 ) and (delta< 1.E-10 ) )
      then pseudo_inv_scalar := 1./ delta
      else pseudo_inv_scalar := 0;
end;

 procedure pseudo_inv_2 (d : sigmaf_type ; var x : sigmaf_type ;
 nu : integer );
  var  det : real; temp : real;
  begin
  if nu > 2 then begin
writeln(' Warning : you may install pseudoinverses for nu >2');
        readln;
                end;
  if nu = 1 then begin
  if abs(d[1,1]) > 1.E-10 then x[1,1] := 1/d[1,1]
  else x[1,1] := 0
 end
    else
     begin
      det := d[1,1]*d[2,2]-d[1,2]*d[2,1] ;
      if abs(det) > 1.E-10
    then begin
 x[1,1] := d[2,2]/det ;
 x[1,2] := -d[1,2]/det;
 x[2,1] := -d[2,1]/det;
 x[2,2] := d[1,1]/det ;
 end
    else
     begin
     temp := d[1,1]*d[1,1]+d[1,2]*d[1,2]+d[2,1]*d[2,1]+d[2,2]*d[2,2];
     if temp > 1.E-10
 then begin
      x[1,1] := d[1,1]/temp ;
      x[1,2] := d[2,1]/temp ;
      x[2,1] := d[1,2]/temp ;
      x[2,2] := d[2,2]/temp ;
      end
 else begin
      x[1,1] := 0;
      x[1,2] := 0;
      x[2,1] := 0;
      x[2,2] := 0;
```

Scanned with OKEN Scanner

```
      end;
      end;
     end;
end; {pseudo_inv_2}

 Procedure Mv_latice_FIR ( nb,ny,nu : integer ; y_current : y_type ;
   u_current : u_type ; lambda : real ;
   var current_latice_par : mv_latice_type ;
   var current_direct_par : mv_direct_type ;
   option_direct_par: boolean );
var
   past : mv_latice_type ; n,i,j,k,m : integer ;
   pssig : sigmaf_type ;temp_eps : u_type ;

begin
 past := current_latice_par ;
 { step 2.1 }
 with current_latice_par do
   begin
    eps[0]  := y_current ;
    epsb[0] := past.epsf[0] ;
    epsf[0] := u_current ;
    for i := 1 to ny do for j := 1 to ny do
     sigma[0][i][j] := lambda * sigma[0][i][j]+ y_current[i]*
   y_current[j] ;
    sigmab[0] := sigmaf[0] ;
    for i := 1 to nu do for j := 1 to nu do
     sigmaf[0][i][j] := lambda*sigmaf[0][i][j] + u_current[i]*
   u_current[j] ;
    gama0[0] := 1 ;
    { Step 2.2 }
    For n := 0 to nb do
     Begin
for i := 1 to nu do for j := 1 to ny do
  alfay[n][j][i] := lambda*alfay[n][j][i] + eps[n][j]*
      epsb[n][i]*pseudo_inv_scalar(gama0[n]) ;
for i := 1 to nu do for j := 1 to nu do
  alfa[n][i][j] := lambda*alfa[n][i][j] + epsb[n][i]*
      epsf[n][j]*pseudo_inv_scalar(gama0[n]) ;
pseudo_inv_2 ( sigmab[n],pssig, nu ) ;
for i := 1 to nu do for j := 1 to ny do
  begin
    ky[n+1][i][j] := 0 ;
    for k := 1 to nu do
      ky[n+1][i][j] := ky[n+1][i][j] - pssig[i,k]*alfay[n][j][k] ;
    end ;
```

Scanned with OKEN Scanner

```
   for i := 1 to nu do for j := 1 to nu do
      begin
kf[n+1][i][j] := 0 ;
for k := 1 to nu do
              kf[n+1][i][j] := kf[n+1][i][j] - pssig[i,k]*alfa[n][k][j] ;
      end ;
   for i := 1 to nu do
      begin
keb[n+1][i] := 0 ;
for k := 1 to nu do
             keb[n+1][i] := keb[n+1][i] - pssig[i,k]*epsb[n][k] ;
      end ;
   pseudo_inv_2 ( sigmaf[n], pssig,nu );
   for i := 1 to nu do for j:= 1 to nu do
      begin
kb[n+1][i][j] := 0 ;
for k := 1 to nu do
     kb[n+1][i][j] := kb[n+1][i][j] - pssig[i,k]*alfa[n][j][k] ;
      end ;
   for i := 1 to nu do for j := 1 to nu do
      begin
kf_time_kb[n+1][i][j] := 0 ;
for k := 1 to nu do
  kf_time_kb[n+1][i][j] := kf_time_kb[n+1][i][j] +
    kf[n+1][i,k]*kb[n+1][k,j] ;
      end ;
   for i := 1 to ny do
      begin
eps[n+1][i] := eps[n][i] ;
for j := 1 to nu do
  eps[n+1][i] := eps[n+1][i] +ky[n+1][j,i] * epsb[n][j] ;
      end ;
   for i := 1 to nu do
      begin
epsf[n+1][i] := epsf[n][i] ;
for j := 1 to nu do
  epsf[n+1][i] := epsf[n+1][i] + kf[n+1][j,i]* epsb[n][j] ;
      end ;
   for i := 1 to nu do
      begin
epsb[n+1][i] := past.epsb[n][i] ;
for j := 1 to nu do
   epsb[n+1][i] := epsb[n+1][i] + past.kb[n+1][j,i]*
past.epsf[n][j] ;
      end ;
   for i := 1 to ny do for j := 1 to ny do
```

Scanned with OKEN Scanner

```
      begin
sigma[n+1][i][j] := sigma[n][i][j] ;
for k := 1 to nu do
  sigma[n+1][i][j] := sigma[n+1][i][j]+ alfay[n][i][k] *
     ky[n+1][k,j] ;
      end;
    for i := 1 to nu do for j := 1 to nu do
       begin
 sigmaf[n+1][i][j] := sigmaf[n][i,j] ;
 for k := 1 to nu do
   sigmaf[n+1][i][j] := sigmaf[n+1][i][j]- sigmaf[n,i,k]*
    kf_time_kb[n+1][j,k] ;
       end ;
    for i := 1 to nu do for j := 1 to nu do
       begin
 sigmab[n+1][i,j] := past.sigmab[n][i,j];
 for k := 1 to nu do
sigmab[n+1][i,j] := sigmab[n+1][i,j] - past.sigmab[n,i,k]*
 past.kf_time_kb[n+1][k,j] ;
       end ;
    gama0[n+1] := gama0[n] ;
    for j := 1 to nu do
      gama0[n+1] := gama0[n+1] + keb[n+1][j] * epsb[n][j] ;
      end ; { for }
  end ; { with }

   { Step 2.3 : Predictors optional recursions }

  if Option_direct_par
   then
     with current_latice_par, current_direct_par do
     { Initialize order recursions }
     begin
for i := 1 to nu do for j := 1 to ny do
   teta[1][1][i][j]:= -ky[1][i][j] ;
for i := 1 to nu do for j := 1 to nu do
begin
  tetaf[1][1][i][j] := -kf[1][i][j] ;
  tetab[1][1][i][j] := -past.kb[1][i][j] ;
end;
for i := 1 to nu do
  kstar[1][1][i] := -keb[1][i] ;
       { Order recursions }
if nb > 1 then
Begin
For n:= 1 to nb-1 do
```

Scanned with OKEN Scanner

```
Begin
{ teta[n+1] }
   for i := 1 to nu do for j := 1 to ny do
     teta[n+1][n+1][i][j] := - ky[n+1][i][j] ;
   for k := 1 to n do
    begin
           teta[n+1][k] := teta[n][k] ;
    for i := 1 to nu do for j := 1 to ny do for m := 1 to nu do
     teta[n+1][k][i][j] := teta[n+1][k][i][j] + tetab[n][k][i][m] *
     ky[n+1][m][j];
    end;
      { tetaf[n+1] }
    for i := 1 to nu do for j := 1 to nu do
    tetaf[n+1][n+1][i][j] := - kf[n+1][i][j] ;
    for k := 1 to n do
      begin
tetaf[n+1][k] := tetaf[n][k] ;
for i := 1 to nu do for j :=1 to nu do for m := 1 to nu do
 tetaf[n+1][k][i][j] := tetaf[n+1][k][i][j] +
 tetab[n][k][i][m]* kf[n+1][m][j] ;
      end ;
      { tetab[n+1] }
    for i := 1 to nu do for j := 1 to nu do
      tetab[n+1][1][i][j] := -past.kb[n+1][i][j] ;
    for i := 1 to nu do begin
      temp_eps[i] := epsb[n][i] ;
      for m := 1 to nu do
temp_eps[i] := temp_eps[i] +epsf[n][m]*past.kb[n+1][m][i] ;
    end ;
    for k := 2 to n+1 do
      begin
tetab[n+1][k] := tetab[n][k-1];
for i := 1 to nu do for j := 1 to nu do for m := 1 to nu do
tetab[n+1][k][i][j] := tetab[n+1][k][i][j] +
    tetaf[n][k-1][i][m]*past.kb[n+1][m][j];
for i := 1 to nu do for j := 1 to nu do
  tetab[n+1][k][i][j] :=tetab[n+1][k][i][j]- temp_eps[j]*
pseudo_inv_scalar(gama0[n])*kstar[n][k-1,i];
      end;
      { kstar[n+1] }
    for i := 1 to nu do
      kstar[n+1][n+1][i] := -keb[n+1][i] ;
    for k := 1 to n do
      begin
kstar[n+1][k] := kstar[n][k] ;
for i:= 1 to nu do for m := 1 to nu do
```

Scanned with OKEN Scanner

```
  kstar[n+1][k][i] := kstar[n+1][k][i] +
tetab[n][k][i][m]*keb[n+1][m];
      end;
     end; { for }
    end; { If }
  end; { with }
end; { Mv_Latice_FIR }


Procedure Initialize_mv_latice( nb,ny,nu : integer ;
 var current_latice_par : mv_latice_type ) ;
var n,i,j,k,m : integer ;

begin
 with current_latice_par do
      for n := 0 to nbmax do
 begin
   for i := 1 to ny do
    begin
    eps[n][i] := 0 ;
    for j := 1 to ny do
       sigma[n][i][j] := 0 ;
    for j := 1 to nu do
     begin
     alfay[n][i][j]:= 0 ;
     ky[n][j][i] := 0 ;
     end;
    end;
   for i := 1 to nu do
    begin
      epsf[n][i] := 0 ;
      epsb[n][i] := 0 ;
      keb[n][i] := 0 ;
      for j := 1 to nu do
begin
  sigmaf[n][i][j] := 0 ;
  sigmab[n][i][j] := 0 ;
  alfa[n][i][j] := 0 ;
  kf[n][i][j] := 0 ;
  kb[n][i][j] := 0 ;
  kf_time_kb[n][i][j] := 0 ;
end;
    end;
  gama0[n] := 0 ;
end; {with}
     end; { Initialize_mv_latice }
```

Scanned with OKEN Scanner

```
  Procedure Mv_cov_update_FIR ( nb,ny,nu : integer ; y_current : y_type;
u_current : u_type ;
var r_M : r_M_type ; var rf_M : rf_M_type;
var fi : fi_type ;
var current_latice_par : mv_latice_type ;
var current_direct_par : mv_direct_type ;
option_order_recursions : boolean );
  var
     past_latice : mv_latice_type ;
     past_direct : mv_direct_type ;
     past_r_M : r_M_type ;
     past_rf_M : rf_M_type ;
     n,k,i,j,m : integer ;

  procedure Compute_Ky (var Kyn : teta_type ; sigb : sigmaf_type;
alfy : alfay_type ; ny,nu : integer );
  var pssig : sigmaf_type ;
  i,j,k : integer ;
  begin
  Pseudo_inv_2(sigb,pssig,nu) ;
  for i := 1 to nu do for j := 1 to ny do
    begin
      Kyn[i][j] := 0;
      for k := 1 to nu do
Kyn[i][j] := Kyn[i][j] - pssig[i,k]*alfy[j,k] ;
    end;
  end; { Compute_Ky }

  procedure Compute_Kf ( var Kfn : sigmaf_type ; sigb,alf : sigmaf_type ;
 nu : integer ) ;
  var pssig : sigmaf_type;
      i,j,k : integer;
  begin
    Pseudo_inv_2(sigb,pssig,nu);
    for i := 1 to nu do for j := 1 to nu do
    begin
      Kfn[i,j] := 0;
      for k := 1 to nu do
Kfn[i,j] := Kfn[i,j]- pssig[i,k]*alf[k,j] ;
    end;
  end; { Compute_Kf }

  procedure Compute_Kb ( var Kbn : sigmaf_type ; sigf,alf : sigmaf_type ;
 nu : integer ) ;
  var pssig : sigmaf_type;
      i,j,k : integer;
```

```
  begin
    Pseudo_inv_2(sigf,pssig,nu);
    for i := 1 to nu do for j := 1 to nu do
    begin
      Kbn[i,j] := 0;
      for k := 1 to nu do
Kbn[i,j] := Kbn[i,j]- pssig[i,k]*alf[j,k] ;
    end;
  end; { Compute_Kf }

  procedure Compute_Keb( var Kebn : u_type;sigb : sigmaf_type ;
 epsbn : u_type; nu : integer );
  var pssig : sigmaf_type ;
  i,j : integer;
  begin
    Pseudo_inv_2 (sigb,pssig,nu);
    for i := 1 to nu do
      begin
Kebn[i] := 0 ;
for j := 1 to nu do
  Kebn[i] := Kebn[i]- pssig[i,j]*epsbn[j] ;
      end;
  end; { Compute_Keb }

  procedure Compute_Kf_times_Kb( var a : sigmaf_type ;b,c : sigmaf_type ;
nu : integer );
  var i,j,k : integer;
  begin
   for i := 1 to nu do for j := 1 to nu do
     begin
      a[i,j] := 0;
      for k := 1 to nu do
a[i,j] := a[i,j] +b[i,k]*c[k,j];
     end;
  end; { Compute_Kf_times_Kb }

  Begin
  { Step 2.1 }
  past_r_M := r_M ;
  past_rf_M := rf_M ;
  past_latice := current_latice_par ;
  for k := 1 to nb do for i := 1 to nu do for j := 1 to ny do
    r_M[k][i][j] := r_M[k][i][j] + fi[k][i] * y_current[j] ;
  for k := 1 to nb do for i := 1 to nu do for j := 1 to nu do
    rf_M[k][i][j] := rf_M[k][i][j] + fi[k][i] * u_current[j] ;
  with current_latice_par do
```

```
begin
  for i := 1 to ny do for j := 1 to ny do
    sigma[0][i][j] := sigma[0][i][j] + y_current[i] * y_current[j] ;
  sigmab[0]:= sigmaf[0] ;
  for i := 1 to nu do for j:= 1 to nu do
    sigmaf[0][i][j] := sigmaf[0][i][j] + u_current[i]*u_current[j] ;
end; { with }

{ Step 2.2 }
If option_order_recursions then
Begin
with current_latice_par,current_direct_par do
Begin
eps[0] := y_current ;
epsb[0] := fi[1] ;
gama0[0] := 1 ;
past_latice.alfa[0] := past_rf_M[1] ;
for i := 1 to ny do for j := 1 to nu do
  alfay[0][i][j] := r_M[1][j][i] ;
Compute_Ky (ky[1],sigmab[0],alfay[0],ny,nu );
Compute_Kf (past_latice.Kf[1],past_latice.sigmab[0],past_latice.alfa[0],nu);
Compute_Kb (past_latice.Kb[1],past_latice.sigmaf[0],past_latice.alfa[0],nu);
Compute_Keb(Keb[1],sigmab[0],epsb[0],nu);
Compute_Kf_times_Kb(past_latice.Kf_time_kb[1],past_latice.Kf[1],
    past_latice.Kb[1],nu) ;
for i := 1 to nu do for j := 1 to ny do
   teta[1][1][i][j] := -Ky[1][i][j] ;
for i := 1 to nu do for j := 1 to nu do
   begin
   past_direct.tetaf[1][1][i][j] := -past_latice.Kf[1][i][j] ;
   tetab[1][1][i][j] := -past_latice.Kb[1][i][j] ;
   end;
for i := 1 to nu do
   kstar[1][1][i] := -Keb[1][i] ;
{ Step 2.2.2 }
 If nb > 1 then
 Begin
 For n:= 0 to nb-2 do
 Begin
 epsb[n+1] := fi[n+2];
 for k := 1 to n+1 do for i:= 1 to nu do for j := 1 to nu do
   epsb[n+1][j] := epsb[n+1][j] - tetab[n+1][k][i][j] * fi[k][i];
 gama0[n+1] := gama0[n];
 for j := 1 to nu do
   gama0[n+1] := gama0[n+1] +Keb[n+1][j]*epsb[n][j] ;
 for i := 1 to ny do for j := 1 to ny do
```

Scanned with OKEN Scanner

```
    begin
    sigma[n+1][i][j] := sigma[n][i][j];
    for k := 1 to nu do
     sigma[n+1][i][j] := sigma[n+1][i][j] + alfay[n][i][k]*ky[n+1][k,j];
    end;
  for i := 1 to nu do for j := 1 to nu do
    begin
    past_latice.sigmaf[n+1][i][j] := past_latice.sigmaf[n][i,j];
    for k := 1 to nu do
    past_latice.sigmaf[n+1][i][j] := past_latice.sigmaf[n+1][i][j]-
      past_latice.sigmaf[n,i,k]*past_latice.Kf_time_Kb[n+1][j,k];
    end;
  for i := 1 to nu do for j := 1 to nu do
    begin
    sigmab[n+1][i,j] :=  past_latice.sigmab[n][i,j];
    for k := 1 to nu do
      sigmab[n+1][i,j] := sigmab[n+1][i,j]-past_latice.sigmab[n,i,k]*
 past_latice.Kf_time_Kb[n+1][k,j];
    end;
  for i := 1 to nu do for j := 1 to nu do
    past_latice.sigmab[n+1][i,j] := sigmab[n+1][i,j]- epsb[n+1][i]*
    epsb[n+1][j]*pseudo_inv_scalar(gama0[n+1]) ;
  for i := 1 to ny do for j := 1 to nu do
    alfay[n+1][i][j] := r_M[n+2][j][i];
  for k := 1 to n+1 do for i := 1 to nu do for j := 1 to ny do for m := 1
    to nu do
   alfay[n+1][j][m] := alfay[n+1][j][m]-r_M[k][i][j]*tetab[n+1][k][i][m];
  for k := 1 to n+1 do for i := 1 to nu do for j := 1 to nu do
    past_direct.tetab[n+1][k][i][j] := past_direct.tetab[n+1][k,i,j]-
Kstar[n+1][k][i]*epsb[n+1][j]*pseudo_inv_scalar(gama0[n+1]) ;
  past_latice.alfa[n+1] := past_rf_M[n+2];
  for k := 1 to n+1 do for i:= 1 to nu do for j := 1 to nu do
     for m := 1 to nu do
    past_latice.alfa[n+1][j][m] := past_latice.alfa[n+1][j][m]-
     past_direct.tetab[n+1][k][i][j]*past_rf_M[k][i][m] ;
  for i := 1 to ny do
    begin
eps[n+1][i] := eps[n][i] ;
for j := 1 to nu do
  eps[n+1][i] := eps[n+1][i] +Ky[n+1][j,i]*epsb[n][j] ;
    end;
   Compute_Ky ( Ky[n+2],sigmab[n+1],alfay[n+1],ny,nu) ;
   Compute_Kf ( past_latice.Kf[n+2],past_latice.sigmab[n+1],
   past_latice.alfa[n+1],nu );
   Compute_Kb ( past_latice.Kb[n+2],past_latice.sigmaf[n+1],
past_latice.alfa[n+1],nu );
```

```
  Compute_Keb ( Keb[n+2],sigmab[n+1],epsb[n+1],nu );
  Compute_Kf_times_Kb( past_latice.Kf_time_Kb[n+2],past_latice.Kf[n+2],
 past_latice.Kb[n+2],nu );
  for i := 1 to nu do for j := 1 to ny do
    teta[n+2][n+2][i][j] := - Ky[n+2][i][j];
  for k := 1 to n+1 do
     begin
     teta[n+2][k] := teta[n+1][k] ;
     for i := 1 to nu do for j := 1 to ny do for m := 1 to nu do
teta[n+2][k][i][j] := teta[n+2][k][i][j] + tetab[n+1][k][i][m]*
 Ky[n+2][m][j] ;
   end;
  for i := 1 to nu do for j := 1 to nu do
     past_direct.tetaf[n+2][n+2][i,j] := - past_latice.Kf[n+2][i,j] ;
  for k := 1 to n+1 do
    begin
     past_direct.tetaf[n+2][k] := past_direct.tetaf[n+1][k] ;
     for i := 1 to nu do for j := 1 to nu do for m := 1 to nu do
past_direct.tetaf[n+2][k][i,j] := past_direct.tetaf[n+2][k][i,j]+
   past_direct.tetab[n+1][k][i][m]*past_latice.Kf[n+2][m][j];
    end;
  for i := 1 to nu do for j := 1 to nu do
     tetab[n+2][1][i][j] := -past_latice.kb[n+2][i][j] ;
  for k := 2 to n+2 do
    begin
      tetab[n+2][k] := past_direct.tetab[n+1][k-1];
      for i := 1 to nu do for j := 1 to nu do for m := 1 to nu do
 tetab[n+2][k][i][j] := tetab[n+2][k][i][j] +
     past_direct.tetaf[n+1][k-1][i,m]*past_latice.Kb[n+2][m][j] ;
   end;
  for i := 1 to nu do
  kstar[n+2][n+2][i] := -Keb[n+2][i];
  for k := 1 to n+1 do
    begin
     kstar[n+2][k] := kstar[n+1][k] ;
     for i := 1 to nu do for m := 1 to nu do
kstar[n+2][k][i] := kstar[n+2][k][i]+tetab[n+1][k][i][m] *
      Keb[n+2][m] ;
    end;
   end; { For }
   end; { If step 2.2.2 }
   end; { with }
 {write_MV_Latpar(nb,nu,ny,y_current[1],
 past_latice,past_direct);
 write_MV_Latpar(nb,nu,ny,y_current[1],
 current_latice_par,current_direct_par);
```

Scanned with OKEN Scanner

```
}
   end; { If step 2.2 }
   for i := nb+1 downto 2 do
     fi[i] := fi[i-1];
  fi[1] := u_current ;

 end; { Mv_cov_update_FIR }

 Procedure Initialize_mv_covar( nb,ny,nu : integer ;
 var r_M : r_M_type; var rf_M : rf_M_type;
 var fi : fi_type ;
 var current_latice_par : mv_latice_type );
  var i,j,k : integer;
  Begin
    for i := 1 to nbmax do for j := 1 to nu do for k := 1 to ny do
      r_M[i][j][k] := 0;
    for i := 1 to nbmax do for j := 1 to nu do for k := 1 to nu do
      rf_M[i][j][k] := 0;
    for i := 0 to nbmax do for j := 1 to nu do
      fi[i,j] := 0;
    for i := 1 to ny do for j := 1 to ny do
      current_latice_par.sigma[0][i][j] := 0;
    for i := 1 to nu do for j := 1 to nu do
      begin
       current_latice_par.sigmaf[0][i][j] := 0;
       current_latice_par.sigmab[0][i][j] := 0;
    end;
  end; { Initialize_mv_covar }

begin
end.
```

Scanned with OKEN Scanner

A.2. Proceduri PASCAL pentru implementarea algoritmilor rapizi pentru modele autoregresive

În această secţiune prezentăm o bibliotecă de proceduri scrise în limbajul Pascal, pentru implementarea următorilor algoritmi de modelare autoregresivă:

Latice_AR Algoritmul 7.3.1 (Tabelul 7.8) "Latice a erorilor de predicţie şi predictorilor" pentru modele monovariabile de tip autoregresiv. Iniţializarea procedurii la momentul $N = 0$ este realizată de procedura *Initialize_latice*.

Fast_Kalman_AR Algoritmul 12.2.1 (Tabelul 12.3) - algoritm Fast Kalman pentru modele monovariabile de tip autoregresiv. Iniţializarea procedurii la momentul $N = 0$ este realizată de procedura *Init_FK*.

FAEST_AR Algoritmul 12.3.1 (Tabelul 12.4) - algoritm FAEST pentru modele monovariabile de tip autoregresiv.

```
procedure Initialize_latice ( na : integer;
      var current_latice_par : latice_par_type);
procedure Latice_AR (na : integer; y: real;
      var current_latice_par : latice_par_type;
      var current_direct_par : direct_par_type;
      Option_direct_par : boolean);

procedure Init_FK( na_true,NinFK :integer;y_t_s : time_serie;
  var ycurrent : real; var fi : teta_par_type;
  var FK_trans_filter :FK_trans_filter_type ;
  var current_latice_par: latice_par_type);

procedure Fast_Kalman_AR ( na: integer ; ycurrent : real ;
   var fi : teta_par_type;
   var FK_trans_filter : FK_trans_filter_type );
procedure FAEST_AR ( na : integer ; ycurrent : real ;
     var  fi : teta_par_type ;
     var FAEST_trans_filter : FAEST_trans_filter_type ;
     var past1 : FAEST_trans_filter_type );


unit fastmodl ;
interface
uses fastgrh,Printer;
const
  na_max = 10;
  Nmaxdim   = 500;

type
```

Scanned with OKEN Scanner

```
array0na_max = array [0..na_max] of real;
array1na_max = array [1..na_max] of real;
teta_par_type =array [1..na_max] of real;
direct_par_type = record
     teta : array [1..na_max] of teta_par_type;
     tetab: array [1..na_max] of teta_par_type;
     Kstar: array [1..na_max] of teta_par_type
  end;
latice_par_type = record
     alfa  : array [0..na_max-1] of real ;
     sigma : array0na_max;
     sigmab: array0na_max ;
     epsf  : array0na_max;
     epsb  : array0na_max;
     gama0 : array0na_max;
     kf    : array1na_max;
     kb    : array1na_max;
     kf_time_kb : array1na_max;
  end;
FK_trans_filter_type = record
 teta  : teta_par_type ;
 tetab : teta_par_type ;
 kstar : teta_par_type ;
 eb    : real ;
 ef    : real ;
 epsf  : real ;
 sigma : real
end;
FAEST_trans_filter_type = record
    teta   : teta_par_type ;
    tetab  : teta_par_type ;
    k_gain : teta_par_type ;
    sigma  : real ;
    sigmab : real ;
    gama   : real ;
    ef     : real ;
    epsf   : real ;
    eb     : real ;
    epsb   : real ;
  end;
const
    const_zero : array0na_max = (0,0,0,0,0,0,0,0,0,0,0,0);
 const1_zero : array1na_max = (0,0,0,0,0,0,0,0,0,0);

procedure Initialize_latice ( na : integer;
      var current_latice_par : latice_par_type);
```

Scanned with OKEN Scanner

```
procedure Latice_AR (na : integer; y: real;
      var current_latice_par : latice_par_type;
      var current_direct_par : direct_par_type;
      Option_direct_par : boolean);

procedure Init_FK( na_true,NinFK :integer;y_t_s : time_serie;
  var ycurrent : real; var fi : teta_par_type;
  var FK_trans_filter :FK_trans_filter_type ;
  var current_latice_par: latice_par_type);

procedure Fast_Kalman_AR ( na: integer ; ycurrent : real ;
   var fi : teta_par_type;
   var FK_trans_filter : FK_trans_filter_type );
procedure FAEST_AR ( na : integer ; ycurrent : real ;
     var  fi : teta_par_type ;
     var FAEST_trans_filter : FAEST_trans_filter_type ;
     var past1 : FAEST_trans_filter_type );

implementation

procedure Initialize_latice ( na : integer;
      var current_latice_par : latice_par_type);
   var k : integer;
   begin
     with current_latice_par do
begin
 for k := 0 to na do  alfa  [k] := 0;
 sigma := const_zero;
 sigmab:= const_zero;
 epsf := const_zero;
 epsb := const_zero;
 gama0 := const_zero;
 kf := const1_zero;
 kb := const1_zero;
 kf_time_kb := const1_zero;
end;{with}
     end; {Initialize_latice}

procedure Latice_AR (na : integer; y: real;
      var current_latice_par : latice_par_type;
      var current_direct_par : direct_par_type;
      Option_direct_par : boolean);
var
   n,k :integer;
   temporar : real;
   past_latice_par : latice_par_type;
```

```
 function pseudo_inv_scalar (delta : real) : real;
begin
if not ((delta > -1.E-10 ) and (delta< 1.E-10 ) )
      then pseudo_inv_scalar := 1./ delta
      else pseudo_inv_scalar := 0;
end;

 begin
    past_latice_par := current_latice_par;
    { step 2.1 Initializing order recursions }
    with current_latice_par do
      begin
epsf [0] := y;
epsb [0] := past_latice_par. epsf [0];
sigma[0] := past_latice_par. sigma[0] + y*y;
sigmab[0]:= past_latice_par. sigma[0];
gama0[0] := 1;
      end;
      { Step 2.2 Lattice order recursions }
      with current_latice_par do
 for n := 0 to na-1 do
  begin
   alfa [n] :=alfa [n]+ epsf[n]*epsb[n]*
      pseudo_inv_scalar(gama0[n]);
   kf [n+1] := -alfa[n]* pseudo_inv_scalar(sigmab[n]);
   kb [n+1] := -alfa[n]* pseudo_inv_scalar(sigma [n]);
   kf_time_kb [n+1] := kf[n+1] * kb[n+1];
   sigma [n+1] := sigma[n]* (1-kf_time_kb [n+1]);
   sigmab[n+1] := past_latice_par. sigmab[n]*
   (1- past_latice_par. kf_time_kb[n+1]);
   epsf[n+1] := epsf[n] + kf[n+1]* epsb[n];
   epsb[n+1] := past_latice_par. epsb[n]+ past_latice_par. epsf[n]*
    past_latice_par. kb[n+1];
   gama0[n+1] := gama0[n]- epsb[n]*epsb[n]*
     pseudo_inv_scalar(sigmab[n]);
   end;
     { Step 2.3 Optional predictors recursions }
     if Option_direct_par
      then
       with current_latice_par, current_direct_par do
{ Initialize order recursions }
begin
 teta [1][1] := kf[1];
 tetab[1][1] := past_latice_par. kb[1];
 Kstar[1][1] := -epsb[0]*pseudo_inv_scalar(sigmab[0]);
{ Order recursions }
```

Scanned with OKEN Scanner

```
 for n := 1 to na-1 do
   begin
    for k := 1 to n do
     teta[n+1][k] := teta[n][k]+kf[n+1] * tetab[n][k];
    teta[n+1][n+1]:= kf[n+1];
    temporar := (epsb[n]+ epsf[n]*past_latice_par. kb[n+1])*
  pseudo_inv_scalar (gama0[n]);
    for k := 2 to n+1 do
     tetab[n+1][k] := tetab[n][k-1]+ past_latice_par.kb[n+1]*
  teta[n][k-1] - temporar*kstar[n][k-1];
    tetab[n+1][1] := past_latice_par.kb[n+1];
    temporar := epsb[n]*pseudo_inv_scalar( sigmab[n] );
    for k := 1 to n do
      Kstar[n+1][k] := Kstar[n][k] - tetab[n][k]*temporar;
    Kstar[n+1][n+1] := - temporar;
   end; {for}
end; { with }
     end; { Latice_AR }

procedure Init_FK( na_true,NinFK :integer;y_t_s : time_serie;
   var ycurrent : real; var fi : teta_par_type;
   var FK_trans_filter :FK_trans_filter_type ;
   var current_latice_par: latice_par_type);
  var t,i : integer;
      current_direct_par : direct_par_type;
  begin
  Initialize_latice( 6, current_latice_par);
  for t := 10 to NinFK do
  begin
     ycurrent := y_t_s [t];
     Latice_AR (na_true + 1, ycurrent, current_latice_par ,
  current_direct_par , true );

  end ; { for }
  { Initialize FK }
  with current_latice_par do
   begin
     FK_trans_filter.eb := epsb[na_true] / gama0[na_true] ;
     FK_trans_filter.ef := epsf[na_true] / gama0[na_true] ;
     FK_trans_filter.sigma := sigma[na_true] ;
     FK_trans_filter.epsf := epsf[na_true] ;
   end; { with }
   FK_trans_filter.teta := current_direct_par.teta[na_true];
   FK_trans_filter.tetab:= current_direct_par.tetab[na_true];
   FK_trans_filter.kstar:= current_direct_par.kstar[na_true];
   ycurrent := y_t_s [NinFK+1] ;
```

Scanned with OKEN Scanner

```
   for i := 1 to na_true+1 do fi[i] :=- y_t_s[NinFK+1-i] ;
   end; { Init_FK }

  procedure Fast_Kalman_AR ( na: integer ; ycurrent : real ;
    var fi : teta_par_type;
    var FK_trans_filter : FK_trans_filter_type );
   var i,iprint  : integer;
       kstar_naplus1 : teta_par_type ;
       temporar : real ;

   begin
     with FK_trans_filter do
       begin
kstar_naplus1[1] := - epsf / sigma ;
for i := 1 to na do
 kstar_naplus1[i+1] := kstar[i] + kstar_naplus1[1] *
teta[i] ;
for iprint := 1 to na+1 do write('kstar(na+1)(',iprint,') =',
   kstar_naplus1[iprint]);
   writeln;
eb := -fi[na+1] ;
for i := 1 to na do
  eb := eb - tetab[i]* fi[i] ;
temporar := 1/( 1 +kstar_naplus1[na+1] * eb ) ;
for i := 1 to na do
  kstar[i] := ( kstar_naplus1[i] - tetab[i] *
  kstar_naplus1[na+1] ) *temporar ;
for i := 1 to na do
  tetab[i] := tetab[i] + kstar[i] * eb ;
ef := ycurrent ;
for i := 1 to na do
  ef := ef - teta[i] * fi[i] ;
for i := 1 to na do
  teta[i] := teta[i] + kstar[i] *ef ;
epsf := ycurrent ;
for i := 1 to na do
  epsf := epsf - teta[i] * fi [i] ;
sigma := sigma + ef* epsf ;
for i := na+1 downto 1 do
  fi[i] := fi[i-1] ;
fi[1] := - ycurrent ;
      end; { with }
    end; { FastKalman_AR }

  procedure FAEST_AR ( na : integer ; ycurrent : real ;
      var  fi : teta_par_type ;
```

```
      var FAEST_trans_filter : FAEST_trans_filter_type ;
      var past1 : FAEST_trans_filter_type );
  var  i: integer ;
       past2 : FAEST_trans_filter_type ;
       k_nplus1 : teta_par_type ;
       gama_nplus1 : real ;
  begin
     past2 := past1 ;
     past1 := FAEST_trans_filter ;
     with FAEST_trans_filter do
begin
k_nplus1 [1] := -past1.ef/ past2.sigma ;
for i := 1 to na do
  k_nplus1[i+1] := past1.k_gain[i] + k_nplus1[1]*past2.teta[i] ;
gama_nplus1 := past1.gama -  past1.ef * k_nplus1[1] ;
eb := - k_nplus1[na+1] * past1.sigmab ;
for i := 1 to na do
k_gain[i] := k_nplus1[i]- k_nplus1[na+1]* past1.tetab[i] ;
gama := gama_nplus1 + eb* k_nplus1[na+1] ;
epsb := eb / (1+gama) ;
ef := ycurrent ;
for i := 1 to na do
   ef := ef - teta[i] *fi[i] ;
epsf := ef / (1+gama) ;
sigma := past1.sigma +ef*epsf ;
sigmab := past1.sigmab +eb*epsb ;
for i := 1 to na do
   begin
    tetab[i] := past1.tetab[i]+k_gain[i]*epsb ;
    teta[i] := past1.teta[i]+k_gain[i]*epsf ;
   end ;{for}
 for i := na+1 downto 1 do
fi[i] := fi[i-1] ;
 fi[1] := -ycurrent ;
       end ; {with}
     end ;

 end.
```

Scanned with OKEN Scanner

A.3. Algoritm latice implementat în MATLAB

În această secţiune este prezentată procedura MATLAB pentru implementarea Algoritmului 10.1.1 (Tabelele 10.5 şi 10.6) - "Latice a erorilor de predicţie şi predictorilor pentru modele multivariabile de tip sumă de convoluţie.

```
%Algoritm latice pentru erorile de predictie

%   1. Initializarea recursiilor in timp ( N = 1 )

a_y(1:ny,1:nb*nu) = zeros(ny,nb*nu);
A_y(1:ny,1:nb*nu) = zeros(ny,nb*nu);

a(1:nu,1:nb*nu) = zeros(nu,nb*nu);
A(1:nu,1:nb*nu) = zeros(nu,nb*nu);

s(1:ny,1:nb*ny) = zeros(ny,nb*ny);
S(1:ny,1:nb*ny) = zeros(ny,nb*ny);

s_f(1:nu,1:nb*nu) = zeros(nu,nb*nu);
s_b(1:nu,1:nb*nu) = zeros(nu,nb*nu);
S_f(1:nu,1:nb*nu) = zeros(nu,nb*nu);
S_b(1:nu,1:nb*nu) = zeros(nu,nb*nu);

Ye(1,1:ny) = zeros(1,ny);

e_f(1,1:nb*nu) = zeros(1,nb*nu);
e_b(1,1:nb*nu) = zeros(1,nb*nu);

Lam=input('Introduceti factorul de deponderare exponentiala [0...1] : ');

optiune = input('Doriti si estimarea parametrilor ? [NU = 0 , DA = 1] : ');

%   2. Recursie in timp
%      2.1. Initializarea recursiei in ordin

for N = 2:Nmax,

     E(1,1:ny)  = (y(:,N))';
     E_f(1,1:nu) = (u(:,N))';
     E_b(1,1:nu) = (u(:,N-1))';
     S(:,1:ny)  = Lam*s(:,1:ny)+y(:,N)*(y(:,N))';
     S_f(:,1:nu) = Lam*s_f(:,1:nu)+u(:,N)*(u(:,N))';
     S_b(:,1:nu) = s_f(:,1:nu);
     G(1,1)  = 1;

%    2.2 Recursie in ordin
```

Scanned with OKEN Scanner

```
    for n = 1:nb

tu = (n-1)*nu+1;
vu = n*nu;
ty = (n-1)*ny+1;
vy = n*ny;
pu = n*nu+1;
ru = (n+1)*nu;
py = n*ny+1;
ry = (n+1)*ny;

pseu = pinv(G(1,n));
if isempty(pseu) pseu = 0; end

KK = Lam*a_y(:,tu:vu);
LL = (E(1,ty:vy))'*E_b(1,tu:vu)*pseu;
A_y(:,tu:vu) = KK + LL;

AA = Lam*a(:,tu:vu);
BB = (E_b(1,tu:vu))'*E_f(1,tu:vu)*pseu;
A(:,tu:vu)   = AA + BB;

CC = E(1,ty:vy);
DD = (A_y(:,tu:vu)*pinv(S_b(:,tu:vu))*(E_b(1,tu:vu))')';
E(1,py:ry)   = CC - DD;

EE = E_f(1,tu:vu);
FF = ((A(:,tu:vu))'*pinv(S_b(:,tu:vu))*(E_b(1,tu:vu))')';
E_f(1,pu:ru) = EE - FF;

GG = e_b(1,tu:vu);
HH = (a(:,tu:vu)*pinv(s_f(:,tu:vu))*(e_f(1,tu:vu))')';
E_b(1,pu:ru) = GG - HH;

II = S(:,ty:vy);
JJ = A_y(:,tu:vu)*pinv(S_b(:,tu:vu))*(A_y(:,tu:vu))';
S(:,py:ry)   = II - JJ;

MM = S_f(:,tu:vu);
NN = (A(:,tu:vu))'*pinv(S_b(:,tu:vu))*A(:,tu:vu);
S_f(:,pu:ru) = MM - NN;

OO = s_b(:,tu:vu);
PP = a(:,tu:vu)*pinv(s_f(:,tu:vu))*(a(:,tu:vu))';
S_b(:,pu:ru) = OO - PP;
```

Scanned with OKEN Scanner

```
 RR = G(1,n);
 QQ = E_b(1,tu:vu)*pinv(S_b(:,tu:vu))*(E_b(1,tu:vu))';
 G(1,n+1)     = RR - QQ;

 SS = (y(:,N))';
 XX = E(1,py:ry);
 Ye(1,py:ry)  = SS - XX;

     end

    if optiune == 1,

T(1:nu,1:ny)   = pinv(S_b(1:nu,1:nu))*(A_y(1:ny,1:nu))';
T_f(1:nu,1:nu) = pinv(S_b(1:nu,1:nu))*A(1:nu,1:nu);
T_b(1:nu,1:nu) = pinv(s_f(1:nu,1:nu))*(a(1:nu,1:nu))';
Ks(1:nu,1)     = pinv(S_b(1:nu,1:nu))*(E_b(1,1:nu))';

for n = 1:nb-1,

    p = n*nu+1;
    q = (n+1)*nu;
    l = n*nu;
    pseug = pinv(G(1,n+1));
    if isempty(pseug) pseug = 0; end

    T(1:q,:)   = [T(1:l,:);zeros(nu,ny)]+[-T_b(1:l,:);eye(nu)]..
                        *pinv(S_b(:,p:q))*(A_y(:,p:q))';
    T_f(1:q,:) = [T_f(1:l,:);zeros(nu)]+[-T_b(1:l,:);eye(nu)]..
                        *pinv(S_b(:,p:q))*A(:,p:q);
    T_b(1:q,:)=[zeros(nu);T_b(1:l,:)]+[eye(nu);-T_f(1:l,:)]..
               *pinv(s_f(:,p:q))*(a(:,p:q))'-[zeros(nu,1);Ks(1:l,1)]*..
               (E_b(1,p:q)-E_f(1,p:q)*pinv(s_f(:,p:q))*(a(:,p:q))')*pseug;
    Ks(1:q,1) = [Ks(1:l,1);zeros(nu,1)]+[-T_b(1:l,:);eye(nu)]..
                       *pinv(S_b(:,p:q))*(E_b(1,p:q))';
end
    end

    a_y = A_y;
    a = A;

    s_f = S_f;
    s_b = S_b;

    e_f = E_f;
    e_b = E_b;
end
```

Scanned with OKEN Scanner

A.4. Algoritm bloc recursiv implementat în MATLAB

În această secţiune este prezentată procedura MATLAB pentru implementarea Algoritmului 11.1.1 din Tabelul 11.1 - algoritm bloc recursiv cu propagarea covarianţelor pentru modele multivariabile de tip sumă de convoluţie.

```
option = input('Doriti evaluarea predictorului Theta(N) ? [0=Nu 1=Da] : ');

% 1. Initializarea recursiilor in timp (N=1)

rM(1:nb*nu,1:ny)   = zeros(nb*nu,ny);
rM_f(1:nb*nu,1:nu) = zeros(nb*nu,nu);
s(1:ny,1:ny)     = zeros(ny);
s_f(1:nu,1:nu)     = zeros(nu);
s_b(1:nu,1:nu)     = zeros(nu);
Fi(1:(nb+1)*nu,1)  = zeros((nb+1)*nu,1);
Ye(1:ny,1)     = zeros(ny,1);
G(1:nb,1)     = zeros(nb,1);

for N = 2:Nmax,
  RM(1:nb*nu,1:ny) = rM(1:nb*nu,:) + Fi(1:nb*nu,1)*(y(:,N))';
  RM_f(1:nb*nu,1:nu) = rM_f(1:nb*nu,:) + Fi(1:nb*nu,1)*(u(:,N))';
  S(1:ny,1:ny)  = s(1:ny,1:ny) + y(:,N)*(y(:,N))';
  S_f(1:nu,1:nu) = s_f(1:nu,1:nu) + u(:,N)*(u(:,N))';
  S_b(1:nu,1:nu) = s_f(1:nu,1:nu);

  if option == 1,
    E(1:ny,1)     = y(:,N);
    E_b(1:nu,1)     = Fi(1:nu,1);
    G(1,1)     = 1;
    a(1:nu,1:nu)    = rM_f(1:nu,:);
    A_y(1:ny,1:nu) = (RM(1:nu,1:ny))';
    T(1:nu,1:ny)    = pinv(S_b(1:nu,1:nu))*(A_y(1:ny,1:nu))';
    t_f(1:nu,1:nu) = pinv(s_b(1:nu,1:nu))*a(1:nu,1:nu);
    T_b(1:nu,1:nu) = pinv(s_f(1:nu,1:nu))*(a(1:nu,1:nu))';
    Ks(1:nu,1)     = pinv(S_b(1:nu,1:nu))*E_b(1:nu,1);

    for n = 1:nb-1,

       pseudo  = pinv(G(n+1,1));
       if isempty(pseudo) == 1, pseudo = 0; end

       E_b(n*nu+1:(n+1)*nu,1)=Fi(n*nu+1:(n+1)*nu,1)-(T_b(1:n*nu,:))'..
                              *Fi(1:n*nu,1);
       G(n+1,1)=G(n,1)-(E_b((n-1)*nu+1:n*nu,1))'*pinv(S_b((n-1)*nu+1:n*nu,:))..
                *E_b((n-1)*nu+1:n*nu,1);
```

Scanned with OKEN Scanner

```
        S(n*ny+1:(n+1)*ny,:)=S((n-1)*ny+1:n*ny,:)-A_y((n-1)*ny+1:n*ny,:)..
                          *pinv(S_b((n-1)*nu+1:n*nu,:))*(A_y((n-1)*ny+1:n*ny,:))';
        s_f(n*nu+1:(n+1)*nu,:)=s_f((n-1)*nu+1:n*nu,:)-(a((n-1)*nu+1:n*nu,:))'..
                             *pinv(s_b((n-1)*nu+1:n*nu,:))*a((n-1)*nu+1:n*nu,:);
        S_b(n*nu+1:(n+1)*nu,:)=s_b((n-1)*nu+1:n*nu,:)-a((n-1)*nu+1:n*nu,:)..
                           *pinv(s_f((n-1)*nu+1:n*nu,:))*(a((n-1)*nu+1:n*nu,:))';
        s_b(n*nu+1:(n+1)*nu,:)=S_b(n*nu+1:(n+1)*nu,:)-E_b(n*nu+1:(n+1)*nu,1)..
                              *(E_b(n*nu+1:(n+1)*nu,1))'*pseudo ;
        A_y(n*ny+1:(n+1)*ny,:)=(RM(n*nu+1:(n+1)*nu,:))'-(RM(1:n*nu,:))'..
                              *T_b(1:n*nu,:);
        t_b(1:n*nu,1:nu)=T_b(1:n*nu,:)-Ks(1:n*nu,1)*(E_b(n*nu+1:(n+1)*nu,1))'..
                           *pseudo ;
        a(n*nu+1:(n+1)*nu,:)=rM_f(n*nu+1:(n+1)*nu,:)-(t_b(1:n*nu,:))'..
                             *rM_f(1:n*nu,:);
        E(n*ny+1:(n+1)*ny,1)=E((n-1)*ny+1:n*ny,1)-A_y((n-1)*ny+1:n*ny,:)..
                            *pinv(S_b((n-1)*nu+1:n*nu,:))*E_b((n-1)*nu+1:n*nu,1);
        Ye(n*ny+1:(n+1)*ny,1)=y(:,N)-E(n*ny+1:(n+1)*ny,1);
        T(1:(n+1)*nu,:)=[T(1:n*nu,:);zeros(nu,ny)]+[-T_b(1:n*nu,:);eye(nu)]..
                        *pinv(S_b(n*nu+1:(n+1)*nu,:))*(A_y(n*ny+1:(n+1)*ny,:))';
        t_f(1:(n+1)*nu,:)=[t_f(1:n*nu,:);zeros(nu)]+[-t_b(1:n*nu,:);eye(nu)]..
                          *pinv(s_b(n*nu+1:(n+1)*nu,:))*a(n*nu+1:(n+1)*nu,:);
        T_b(1:(n+1)*nu,:)=[zeros(nu);t_b(1:n*nu,:)]+[eye(nu);-t_f(1:n*nu,:)]..
                         *pinv(s_f(n*nu+1:(n+1)*nu,:))*(a(n*nu+1:(n+1)*nu,:))';
        Ks(1:(n+1)*nu,1)=[Ks(1:n*nu,1);zeros(nu,1)]+[-T_b(1:n*nu,:);eye(nu)]..
                         *pinv(S_b(n*nu+1:(n+1)*nu,:))*E_b(n*nu+1:(n+1)*nu,1);
    end
  end

  for i = nb+1:-1:2, Fi((i-1)*nu+1:i*nu,1) = Fi((i-2)*nu+1:(i-1)*nu,1); end
  Fi(1:nu,1) = u(:,N);

  rM   = RM;
  rM_f = RM_f;
  s    = S(1:ny,1:ny);
  s_f(1:nu,1:nu) = S_f;
  s_b  = S_b;
  t_b  = T_b;

end
```

Scanned with OKEN Scanner

Bibliografie

[1] H. Akaike. Block Toeplitz matrix inversion. *SIAM Journal of Applied Mathematics*, 24:234–241, 1973.

[2] A. Benveniste and C. Chaure. AR and ARMA identification algorithms of Levinson type: an innovation approach. *IEEE Transactions on Automatic Control*, AC–26:1243–1261, 1981.

[3] Y. Bistritz, H. Lev-Ari, and T. Kailath. Immitance–domain Levinson algorithms. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 253–256, Tokio, Japan, 1986.

[4] G. Carayannis, D. Manolakis, and N.Kalouptsidis. A fast sequential algorithm for least–squares filtering and prediction. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–31:1394–1402, 1983.

[5] J.M. Cioffi. Windowed fast transversal adaptive filter algorithms with normalization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–33:607–625, 1985.

[6] J.M. Cioffi and T. Kailath. Fast, recursive–least squares transversal filter for adaptive filtering. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–32:304–337, 1984.

[7] G. Cybenko. The numerical stability of the Levinson–Durbin algorithm for Toeplitz systems of equations. *SIAM Journal on Scientific Statistic. Computation*, 1:303–319, 1980.

[8] G. Cybenko. Locations of zeros of predictor polynomials. *IEEE Transactions on Automatic Control*, AC-27:235–237, 1982.

[9] P. Delsarte and Y. Genin. The split Levinson algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–34:470–478, 1986.

[10] P. Delsarte and Y. Genin. On the splitting of classical algorithms in linear prediction theory. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–35:645–653, 1987.

[11] P. Delsarte, Y. Genin, and Y. Kamp. A generalization of the Levinson algorithm for hermitian Toeplitz matrices with any rank profile. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–33:964–971, 1985.

[12] P. Delsarte, Y. Genin, and Y. Kamp. Stability margins of linear prediction polynomials. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 3003–3005, Tokyo, Japan, 1986.

Scanned with OKEN Scanner

[13] C.J. Demeure and L.L. Scharf. Linear statistical models for stationary sequences and related algorithms for Cholesky factorization of Toeplitz matrices. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–35:29–42, 1987.

[14] J. Durbin. The fitting of time series models. *Rev. Inst. Int. Statist.*, 28:232–244, 1960.

[15] B. Friedlander. Lattice filters for adaptive processing. *Proceedings of IEEE*, 70:829–867, 1982.

[16] B. Friedlander. Lattice methods for spectral estimation. *Proceedings IEEE*, 70:990–1017, 1982.

[17] B. Friedlander. Instrumental Variable methods for ARMA spectral estimation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31:404–415, 1983.

[18] B. Friedlander. Lattice implementation of some recursive parameter estimation algorithms. *International Journal of Control*, 37:661–681, 1983.

[19] M. Gevers and V. Wertz. A d–step predictor in lattice and ladder form. *IEEE Transactions on Automatic Control*, AC–28:465–476, 1983.

[20] J. de Goojier and P. Stoica. Min-max optimal instrumental variable estimation method for multivariable time–series systems. *International Journal of Control*, 50:955–976, 1989.

[21] D. Graupe, V.K. Jain, and J. Salahi. A comparative study of various least squares identification algorithms. *Automatica*, 16:663–681, 1980.

[22] C. Gueguen. Linear prediction in the singular case and the stability of eigen modes. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 881–884, Atlanta, GA, 1981.

[23] M.L. Honig and D.G. Messerchmitt. *Adaptive filters.* Kluwer Academic, New York, 1984.

[24] Y. Inouye. Approximation of multivariable linear systems with impulse response and autocorrelation sequences. *Automatica*, 19:265–277, 1983.

[25] Y. Inouye. Modeling of multichannel time series and extrapolation of matrix–valued autocorrelation sequences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–31:45–55, 1983.

[26] I.S. Iohvidov. *Hankel and Toeplitz matrices and forms.* Birkhäuser, Boston MA, 1982.

[27] F. Itakura and S. Saito. Digital filtering techniques for speech analysis and synthesis. In *Proceedings of 7'th Int. Congr. on Acoustics*, 1971.

[28] E. Karlsson. *Least squares ARMA modelling of linear time–varying systems.* PhD dissertation, Georgia Institute of Technology, 1987.

[29] H. Krishna. New split Levinson, Schur and lattice algorithms for digital signal processing. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 1640–1643, 1988.

Scanned with OKEN Scanner

[30] H. Krishna and S. Morgara. The Levinson recurrence and fast algorithms for solving Toeplitz systems of linear equations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–35:839–848, 1987.

[31] S. Lang and J. McClellan. A simple proof of stability for all-pole linear prediction models. *Proceedings IEEE*, 67:860–861, 1979.

[32] D.T. Lee. *Cannonical Ladder form realization and fast estimation algorithms.* PhD dissertation, Stanford University, Department of Electrical Engineering, 1981.

[33] D.T. Lee, M. Morf, and B. Friedlander. Recursive least squares ladder estimation algorithms. *IEEE Transactions on Circuits and Systems*, CAS–28:467–481, 1981.

[34] D.T. Lee, M. Morf, and B. Friedlander. Recursive ladder algorithms for ARMA modeling. *IEEE Transactions on Automatic Control*, AC–27:753–764, 1982.

[35] H. Lev-Ari. *Nonstationary lattice–filter modeling.* PhD dissertation, Stanford University, Stanford, California, Department of Electrical Engineering, 1983.

[36] H. Lev-Ari and T. Kailath. On generalized Schur and Levinson – Szegö algorithms for quasistationary processes. In *IEEE Control and Decision Conference*, pages 1077–1080, 1981.

[37] H. Lev-Ari and T. Kailath. Lattice filter parametrization and modeling of nonstationary processes. *IEEE Transactions on Information Theory*, IT–30:2–16, 1984.

[38] H. Lev-Ari, T. Kailath, and J. Cioffi. Least squares adaptive lattice and transversal filters: A unified geometric theory. *IEEE Transactions on Information Theory*, IT–30:222–236, 1984.

[39] N. Levinson. The Wiener RMS (root mean square) error criterion in filter design and prediction. *In N. Wiener: Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, John Wiley & Sons, New York 1949.

[40] S. Li and B. Dickinson. Application of the lattice filter to robust estimation of AR and ARMA models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-36:502–512, 1988.

[41] F. Ling. *Rapidly convergent adaptive filtering algorithms with applications to adaptive filtering and channel estimation.* PhD dissertation, Northeastern University, Boston, Massachusetts, 1984.

[42] F. Ling, D. Manolakis, and J. Proakis. Numerically robust least squares lattice–ladder algorithms with direct updating of the reflection coefficients. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34:837–845, 1986.

[43] L. Ljung, M. Morf, and D. Falconer. Fast calculation of gain matrices for recursive estimation schemes. *International Journal of Control*, 27:1–19, 1978.

[44] S. Ljung. *Fast algorithms for integral equations and least squares identification problems.* PhD dissertation, Linkoping, Sweden, 1983.

Scanned with OKEN Scanner

[62] S. Pombra, H. Lev-Ari, and T. Kailath. Levinson and Schur algorithms for Toeplitz matrices with singular minors. *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 1643–1646, 1988.

[63] B. Porat. *Contributions to the theory and applications of lattice filters.* PhD dissertation, Stanford University, California, 1982.

[64] B. Porat. The use of prior information in normalized lattice algorithms. *IEEE Transactions on Automatic Control*, AC–27:989–991, 1982.

[65] B. Porat, B. Friedlander, and M. Morf. Square root covariance ladder algorithms. *IEEE Transactions on Automatic Control*, AC–27:813–829, 1982.

[66] B. Porat and T. Kailath. Normalized lattice algorithms for least squares FIR system identification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–31:122–128, 1983.

[67] C. Rialan and L. Scharf. Fast algorithms for computing QR and Cholesky factors of Toeplitz operators. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–36:1740–1748, 1988.

[68] C. Rialan and L. Sharf. Finite precision analysis of the lattice and the Schur algorithms for the autocorrelation method of linear prediction. *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 2252–2255, 1988.

[69] C. Samson. A unified treatment of fast algorithms for identification. *International Journal of Control*, 35:909–934, 1982.

[70] C. Samson and V.U. Reddy. Fixed–point error analysis of the normalized ladder algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–31:1177–1191, 1983.

[71] M.J. Shensa. Recursive least squares lattice algorithms– a geometrical approach. *IEEE Transactions on Automatic Control*, AC–26:695–703, 1981.

[72] T. Söderström and P. Stoica. New and simple proofs for stability of linear prediction models. *Bul. Inst. Politehnic Bucureşti*, 43:95–100, 1981.

[73] T. Söderström and P. Stoica. On the stability of dynamic models obtained by least squares identification. *IEEE Transactions on Automatic Control*, AC-26:575–577, 1981.

[74] T. Söderström and P. Stoica. *Stability properties of autoregressive models fitted to multivariable time series.* Report UPTEC 81 55R, Inst. of Technology, Uppsala University, Uppsala, Suedia, 1981.

[75] T. Söderström and P. Stoica. *Instrumental Variable Methods for System Identification.* Springer, New York, 1983.

[76] T. Söderström and P. Stoica. *System Identification.* Prentice–Hall, London, 1989.

[77] P. Stoica. On a procedure for testing the orders of time series. *IEEE Transactions on Automatic Control*, AC-26:572–573, 1981.

Scanned with OKEN Scanner

[93] J. Turner. Use of digital lattice structure in estimation and filtering. In M. Kunt and F. de Coulon, editors, *Signal processing: Theories and applications.* North Holland Publishing Company, 1980.

[94] A. Vieira. *Matrix orthogonal polynomials with applications to autoregressive modelling and ladder forms.* PhD dissertation, Stanford University, Stanford, California, 1977.

[95] A. Vieira and T.Kailath. On another aproach to the Schur–Cohn criterion. *IEEE Transactions on Circuits and Systems*, CAS–24:218–220, 1977.

[96] P. Whittle. On the fitting of multivariate autoregressions and the approximate canonical factorization of a spectral density matrix. *Biometrika*, 50:129–134, 1963.

[97] R.A. Wiggins and E.A. Robinson. Recursive solution to the multichannel filtering problem. *Journal of Geophysical Research*, 70:1885–1891, 1966.

[98] F. Yarman-Yural. An efficient and oreder–recursive algorithm for estimating stationary ARMA models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–33:1054–1058, 1985.

[99] S. Zohar. Toeplitz matrix inversion – the algorithm of W.F. Trench. *Journal of ACM*, 16:592–601, 1969.

[100] S. Zohar. The solution of a Toeplitz set of linear equations. *Journal of ACM*, 21:272–278, 1974.

Scanned with OKEN Scanner

PETRE STOICA, IOAN TĂBUŞ, MIHAI TERTIŞCO, **Metode rapide pentru modelarea semnalelor şi sistemelor (Fast Methods for Signal and System Modelling)**, Editura Academiei Române, Bucureşti, 1997, 224 p.

Scanned with OKEN Scanner

ABSTRACT

The book presents in a unified manner some of the most computationally efficient and widely used fast methods for signal and system modelling. In the last decades there has been an explosive development in the field of fast modelling algorithms. However, the diversity of approaches and their complexity have made it difficult to follow and capture the algorithmic similarity of different solutions and even more difficult to understand the deep connections between these methods and system and signal theory.

An algebraic approach is used in this book for unifying the derivation of the fast algorithms, exploiting the regular structuring of the solution when decomposing the computations along the time coordinate (time recursiveness) or along the model complexity coordinate (order recursiveness). Problems that are fundamental to signal and system theory, such as stability and positivity are shown to be solved very efficiently and insightfully by the fast modelling algorithms.

The book is structured in three main parts: the first part encompasses the recursive–in–order modelling methods (Chapters 1-6), having the Levinson algorithm as a prototype; the second part (Chapters 7-11) addresses the structures that are recursive both in order and in time, having the lattice structure as a prototype; the third part (Chapters 12-13) is devoted to the fast recursive–in–time algorithms, having their roots in the Fast Kalman algorithm. For each class of methods, the algorithms are specialized to different types of models: autoregressive, autoregressive moving average, multivariate convolution sum and difference equation models. The appendix contains procedures written in PASCAL and MATLAB for the implementation of some fast modelling algorithms. This will facilitate the use of the fast algorithms in various applications, without the need to completely understand the theoretical foundation of the algorithmic structures.

The book addresses the researchers interested in the applications and the theoretical aspects of modelling when the number of parameters is relatively large (tens or hundreds). The main goal of the book is to constitute a reference monograph for researchers in the fields of biology, ecology, econometrics, mathematics, signal processing, automatic control and electronics with interest in the aforementioned modelling problem.

Contents

Part one. Fast Recursive–in–Order Algorithms

1. Levinson–Durbin Algorithm for AR Models 11
   1.1. AR models . . . 11
   1.2. Levinson-Durbin algorithm . . . 14
   1.3. Analysis of model stability . . . 17
   1.4. Applications of the Levinson– Durbin algorithm . . . 22
      1.4.1. Factorization of the inverse covariance matrix . . . 22
      1.4.2. Testing the positivity of Toeplitz matrices . . . 23
      1.4.3. Testing model stability . . . 24

2. Delsarte–Genin and Schur Algorithms for AR Models 27
   2.1. Delsarte–Genin algorithm . . . 27
   2.2. Schur algorithm . . . 31

3. Levinson Algorithm for ARMA Models 39
   3.1. AR–plus–noise and ARMA models . . . 39
   3.2. Levinson type algorithm . . . 42
   3.3. Similarities between the Levinson algorithm and the Levinson–Durbin algorithm 46

4. Whittle–Wiggins–Robinson Algorithm for Multivariate AR Models 49
   4.1. Multivariate AR models . . . 49
   4.2. Whittle–Wiggins-Robinson algorithm . . . 51
   4.3. Analysis of model stability . . . 56
   4.4. Applications of the Wittle–Wiggins–Robinson algorithm . . . 59
      4.4.1. Factorization of the inverse covariance matrix . . . 60
      4.4.2. Testing the positivity of block Toeplitz matrices . . . 61
      4.4.3. Testing model stability . . . 61

5. Levinson Algorithm for Multivariate Models 63
   5.1. Convolution sum models . . . 63
   5.2. Levinson algorithm . . . 65

6. Fast Algorithms for Difference Equation Models 71
   6.1. Difference equation models . . . 71
   6.2. Analysis of model stability . . . 73

Scanned with OKEN Scanner

Part two. Fast Recursive–in–Order–and–in–Time Algorithms

7. Lattice Algorithm for AR Models 77
- 7.1. Lattice parameterization of AR models . . . 77
- 7.2. Prediction error lattice algorithm . . . 86
- 7.3. Lattice algorithm for predictors . . . 95
  - 7.3.1. Derivation of time recursive equations . . . 95
  - 7.3.2. Derivation of order recursive equations . . . 100
  - 7.3.3. Variants of prediction error lattice and predictor lattice algorithms . . . 105
  - 7.3.4. Exact initialization of lattice algorithms . . . 105

8. Normalized Lattice Algorithms for AR Models 111
- 8.1. Normalized prediction error lattices . . . 111
- 8.2. Lattice algorithm for normalized predictors . . . 117
- 8.3. Normalized lattice parameterization of AR models . . . 122

9. Lattice Algorithm for ARMA Models 125
- 9.1. Lattice algorithm for prediction errors . . . 125
- 9.2. Conversion of lattice parameters to AR parameters . . . 136

10. Lattice Algorithm for Multivariate Models 139
- 10.1. Lattice algorithm for prediction errors . . . 139
- 10.2. Lattice algorithm for predictors . . . 154

11. Least Squares Algorithm with Covariance Updating 157
- 11.1. Least Squares algorithm with covariance updating . . . 157
- 11.2. Data windows for tracking time–varying parameters . . . 161

Part three. Fast Recursive–in–Time Algorithms

12. Recursive–in–Time Algorithms for AR Models 169
- 12.1. Conventional recursive–in–time LS method . . . 169
- 12.2. Fast Kalman algorithm . . . 172
- 12.3. FAEST algorithm . . . 174
- 12.4. "Fast Transversal Filter" algorithm . . . 177
- 12.5. Normalized "Fast Transversal Filter" algorithm . . . 179

13. "Fast Transversal Filter" Algorithm for Multivariate Models 185
- 13.1. Time recursive algorithms for multivariate models . . . 185
- 13.2. "Fast Transversal Filter" algorithm . . . 185

Appendices: Procedures for the Implementation of Fast Modelling Algorithms 189

References . . . 215

Scanned with OKEN Scanner

ISBN 973-27-0587-6

Lei

Scanned with OKEN Scanner